Practitioner's Docket No. NVIDP064/P000286 *PATENT*

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Christopher D.S. Donham et al.

Application No.: 10/006,551 Group No.: 2628
Filed: 11/30/2001 Examiner: Amin, Jwalant B.
For: SYSTEM, METHOD AND COMPUTER PROGRAM PRODUCT FOR USING TEXTURES AS INSTRUCTIONS FOR GRAPHICS PROCESSING

Mail Stop Appeal Briefs – Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF
(PATENT APPLICATION--37 C.F.R. § 41.37)

1. This brief is in furtherance of the Notice of Appeal, filed in this case on 12/19/2007.

2. STATUS OF APPLICANT

This application is on behalf of other than a small entity.

3. FEE FOR FILING APPEAL BRIEF

Pursuant to 37 C.F.R. § 41.20(b)(2), the fee for filing the Appeal Brief is:

other than a small entity	$510.00
Appeal Brief fee due	**$510.00**

4. EXTENSION OF TERM

The proceedings herein are for a patent application and the provisions of 37 C.F.R. § 1.136 apply.

Applicant petitions for an extension of time under 37 C.F.R. § 1.136 (fees: 37 C.F.R. § 1.17(a)(1)-(5)) for one month:

Fee: $120.00

If an additional extension of time is required, please consider this a petition therefor.

5. TOTAL FEE DUE

The total fee due is:

Appeal brief fee	$0.00 (previously paid on 04/30/2004)
Extension fee (if any)	$120.00
TOTAL FEE DUE	**$120.00**

6. FEE PAYMENT

Authorization is hereby made to charge the extension fee amount of $120.00 to Deposit Account No. 50-1351 (Order No. NVIDP064).

Applicant believes that the Appeal Brief fee is not due in connection with the filing of this paper because the Appeal Brief fee was paid with a previous submission. However, the Commissioner is authorized to charge any additional fees that may be due (e.g. for any reason including, but not limited to, fee changes, etc.) to Deposit Account No. 50-1351 (Order No. NVIDP064).

7. FEE DEFICIENCY

If any additional extension and/or fee is required, and if any additional fee for claims is required, charge Deposit Account No. 50-1351 (Order No. NVIDP064).

Date: March 19, 2008

/KEVINZILKA/
Signature of Practitioner
Kevin J. Zilka
Zilka-Kotab, PC
P.O. Box 721120
San Jose, CA 95172

Reg. No.: 41,429
Tel. No.: 408-971-2573
Customer No.: 75359

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:	
Donham et al.	Group Art Unit: 2628
Application No.: 10/006,551	Examiner: Amin, Jwalant B.
Filed: 11/30/2001	Atty Docket No.: NVIDP064/P000286
	Date: 03/19/2008
For: SYSTEM, METHOD AND COMPUTER PROGRAM PRODUCT FOR USING TEXTURES AS INSTRUCTIONS FOR GRAPHICS PROCESSING	

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

ATTENTION: Board of Patent Appeals and Interferences

APPEAL BRIEF (37 C.F.R. § 41.37)

This brief is in furtherance of the Notice of Appeal, filed in this case on 12/19/2007.

The fees required under § 1.17, and any required petition for extension of time for filing this brief and fees therefor, are dealt with in the accompanying TRANSMITTAL OF APPEAL BRIEF.

This brief contains these items under the following headings, and in the order set forth below (37 C.F.R. § 41.37(c)(i)):

I REAL PARTY IN INTEREST
II RELATED APPEALS AND INTERFERENCES
III STATUS OF CLAIMS
IV STATUS OF AMENDMENTS

V SUMMARY OF CLAIMED SUBJECT MATTER

VI GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

VII ARGUMENT

VIII CLAIMS APPENDIX

IX EVIDENCE APPENDIX

X RELATED PROCEEDING APPENDIX

The final page of this brief bears the practitioner's signature.

I REAL PARTY IN INTEREST (37 C.F.R. § 41.37(c)(1)(i))

The real party in interest in this appeal is NVIDIA Corporation.

II RELATED APPEALS AND INTERFERENCES (37 C.F.R. § 41.37(c) (1)(ii))

With respect to other prior or pending appeals, interferences, or related judicial proceedings that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, prior appeals were noted on 03/09/2004 and 11/18/2005 in the present application.

A Related Proceedings Appendix is appended hereto.

III STATUS OF CLAIMS (37 C.F.R. § 41.37(c) (1)(iii))

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1-30

B. STATUS OF ALL THE CLAIMS IN APPLICATION

1. Claims withdrawn from consideration: None
2. Claims pending: 1-30
3. Claims allowed: None
4. Claims rejected: 1-30
5. Claims cancelled: None

C. CLAIMS ON APPEAL

The claims on appeal are: 1-30

See additional status information in the Appendix of Claims.

IV STATUS OF AMENDMENTS (37 C.F.R. § 41.37(c)(1)(iv))

As to the status of any amendment filed subsequent to final rejection, the Amendment submitted on 12/14/2004 was not entered by the Examiner.

V SUMMARY OF CLAIMED SUBJECT MATTER (37 C.F.R. § 41.37(c)(1)(v))

With respect to a summary of Claim 1, as shown in Figures 3A and 5 et al., a method for execution with a system including a tangible computer readable medium for retrieving instructions from video memory utilizing a texture module in a graphics pipeline comprises sending an instruction request to video memory (e.g. see item 304 of Figure 3A) utilizing a texture module (e.g. see item 301 of Figure 3A) in a graphics pipeline (e.g. see item 502 of Figure 5, etc.). In addition, the method comprises, in response to the instruction request, receiving the instructions from the video memory utilizing the texture module in the graphics pipeline (e.g. see item 504 of Figure 5, etc.). See, for example, page 6, lines 4-8 et al.

With respect to a summary of Claim 24, as shown in Figures 3A and 5 et al., a computer program product embodied on a tangible computer readable medium for retrieving instructions from video memory utilizing a texture module in a graphics pipeline comprises computer code for sending an instruction request to video memory (e.g. see item 304 of Figure 3A). Further, a texture module (e.g. see item 301 of Figure 3A) in a graphics pipeline sends the instruction request to the video memory (e.g. see item 502 of Figure 5, etc.). In addition, the computer program product comprises computer code for receiving instructions from the video memory in response to the instruction request utilizing the texture module in the graphics pipeline (e.g. see item 504 of Figure 5, etc.). See, for example, page 6, lines 4-8 et al.

With respect to a summary of Claim 25, as shown in Figures 3A and 5 et al., a system including a tangible computer readable medium for retrieving instructions from video memory utilizing a texture module in a graphics pipeline means for (e.g. see item 301 of Figure 3A) sending an instruction request to video memory (e.g. see item 304 of Figure 3A). Further, a texture module (e.g. see item 301 of Figure 3A) in a graphics pipeline sends the instruction request to the video memory (e.g. see item 502 of Figure 5, etc.). In addition, the system comprises means for (e.g. see item 301 of Figure 3A) receiving instructions from the video memory in response to the instruction request (e.g. see item 504 of Figure 5, etc.). See, for example, page 6, lines 4-8 and 16-17 et al.

With respect to a summary of Claim 26, as shown in Figures 3A and 5 et al., a texture sub-system including a tangible computer readable medium for retrieving instructions from video memory and capable of carrying out a method. The method comprising sending an instruction request to video memory (e.g. see item 304 of Figure 3A). Further, a texture module (e.g. see item 301 of Figure 3A) sends the instruction request to the video memory. In addition, the method comprises receiving instructions from the video memory in response to the instruction request (e.g. see item 504 of Figure 5, etc.). See, for example, page 6, lines 4-8 et al.

With respect to a summary of Claim 27, as shown in Figures 3A and 4 et al., a data structure is stored in a frame buffer of a graphics processor including a tangible computer readable medium. Further, the data structure allows for the retrieval of instructions. Additionally, a texture module (e.g. see item 301 of Figure 3A) coupled to the data structure sends an instruction request to video memory (e.g. see item 304 of Figure 3A). In addition, the data structure comprises an instruction object (e.g. see item 402 of Figure 4, etc.) stored in the frame buffer for retrieval in response to the instruction request utilizing the texture module in the graphics processor. See, for example, page 6, lines 4-8 and 16-17; and page 11, lines 13-15 et al.

With respect to a summary of Claim 28, as shown in Figures 3A and 7, a method for execution with a system including a tangible computer readable medium for retrieving instructions from video memory comprises receiving a plurality of preliminary instructions (e.g. see item 702 of Figure 7, etc.) from a rasterizer module (e.g. see item 300 of Figure 3A, etc.) utilizing a texture module (e.g. see item 301 of Figure 3A, etc.) coupled to the rasterizer. The method further comprises sending an instruction request to video memory (e.g. see item 304 of Figure 3A, etc.). Further, the texture module sends the instruction request to the video memory.

Furthermore, the method comprises receiving additional instructions from the video memory in response to the instruction request utilizing the texture module. Additionally, the method comprises caching the additional instructions on the texture module (e.g. see item 303 of Figure 3A, etc.). Further still, the method comprises sending a texture request to video memory utilizing the texture module in accordance with the additional instructions. Additionally, the method comprises receiving texture information from the video memory in response to the

texture request utilizing the texture module. In addition, the method comprises caching the texture information on the texture module.

Further still, the method comprises repeating the steps from sending an instruction request to video memory to caching the texture information on the texture module (e.g. see item 712 of Figure 7, etc.) in accordance with the additional instructions. See, for example, page 6, lines 4-8 and 24-27; page 9, lines 26-30; and page 12, line 26-page 13, line 2 et al.

With respect to a summary of Claim 29, as shown in Figures 6 and 7, a method for execution with a system including a tangible computer readable medium for retrieving instructions from video memory comprises receiving a plurality of preliminary instructions from a rasterizer module (e.g. see item 650 of Figure 6, etc.) utilizing a shader module (e.g. see item 601 of Figure 6, etc.) coupled to the rasterizer (e.g. see item 702 of Figure 7, etc.). The method further comprises sending an instruction request to video memory (e.g. see item 604 of Figure 6, etc.). Additionally, a texture module (e.g. see item 602 of Figure 6, etc.) coupled to the shader module sends the instruction request to the video memory.

Furthermore, the method comprises receiving additional instructions from the video memory in response to the instruction request utilizing the texture module. Additionally, the method comprises caching the additional instructions on the texture module. Further still, the method comprises sending a texture request to video memory utilizing the texture module in accordance with the additional instructions. Additionally, the method comprises receiving texture information from the video memory in response to the texture request utilizing the texture module. Furthermore, the method comprises caching the texture information on the texture module. Further still, the method comprises processing a plurality of pixels with the texture information utilizing the shader module in accordance with the additional instructions (e.g. see item 710 of Figure 7, etc.).

Further still, the method comprises repeating the steps from sending an instruction request to video memory to processing a plurality of pixels with texture information utilizing the shader module (e.g. see item 712 of Figure 7, etc.) in accordance with the additional instructions. See, for example, page 6, lines 4-8; page 7, lines 7-15; and page 9, lines 26-30 et al.

With respect to a summary of Claim 30, as shown in Figure 3A, a method for execution with a system including a tangible computer readable medium for retrieving instructions from video memory utilizing a cache in a graphics pipeline comprises sending an instruction request to video memory (e.g. see item 304 of Figure 3A, etc.) in a graphics pipeline (e.g. see item 303 of Figure 3A, etc.). Further, a cache in the graphics pipeline sends the instruction request to the video memory. Furthermore, the method comprises receiving instructions from the video memory in response to the instruction request for storage in the cache in the graphics pipeline. See, for example, page 6, lines 4-8 and 16-17; page 7, lines 4-5; and page 9, lines 26-30 et al.

Of course, the above citations are merely examples of the above claim language and should not be construed as limiting in any manner.

VI GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL (37 C.F.R. § 41.37(c)(1)(vi))

Following, under each issue listed, is a concise statement setting forth the corresponding ground of rejection.

Issue # 1: The Examiner has rejected Claims 1-12, 18-21, 24-28, and 30 under 35 U.S.C. 103(a) as being unpatentable over Rivard (U.S. Patent No. 5,987,567), in view of Wang (U.S. Patent No. 5,831,640).

Issue #2: The Examiner has rejected Claims 13-17, 22, 23, and 29 under 35 U.S.C. 103(a) as being unpatentable over Rivard (U.S. Patent No. 5,987,567), in view of Wang (U.S. Patent No. 5,831,640), and further in view of Applicant Admitted Prior Art (AAPA).

VII ARGUMENT (37 C.F.R. § 41.37(c)(1)(vii))

The claims of the groups noted below do not stand or fall together. In the present section, appellant explains why the claims of each group are believed to be separately patentable.

Issue # 1:

The Examiner has rejected Claims 1-12, 18-21, 24-28, and 30 under 35 U.S.C. 103(a) as being unpatentable over Rivard (U.S. Patent No. 5,987,567), in view of Wang (U.S. Patent No. 5,831,640).

Group #1: Claims 1-12, 21, 24, and 27

To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art and not based on appellant's disclosure. *In re Vaeck*, 947 F.2d 488, 20 USPQ2d 1438 (Fed.Cir.1991).

With respect to the first element of the *prima facie* case of obviousness, the Examiner has argued that "[o]ne of ordinary skill in [the] art... would have expected [appellant's] invention to perform equally well with Rivard's reference that teaches to send and receive information/instruction from the texture mapping stage and texel cache system to the DRAM because using [these] components together will also result in sending and receiving instructions to and from DRAM." To the contrary, appellant respectfully asserts that it would not have been obvious to modify the teaching of Rivard, especially in view of the vast evidence to the contrary.

Appellant respectfully points out that Rivard teaches that "[b]ecause memory request generator 1020 is between cache tag blocks 1010, 1015 and cache data store 1030, generator 1020 can

perform DRAM 655 memory requests before the address and instruction information reach cache data store and memory data resolver 1030" (Col. 6, lines 62-66 - emphasis added). In addition, Rivard teaches that "[o]nce memory requests are generated, there is, depending on the DRAM design, a constant latency of about five to ten clock cycles (or possibly more) which includes time for exiting and reentering the graphics pipeline hardware, to effect a page hit," and "[t]herefore, graphics accelerator system 635 includes pipeline latency elements 1025 to coordinate arrival of the memory data and of the associated instructions at cache data store and memory data resolver 1030" (Col. 6, line 66-Col. 7, line 7 - emphasis added). Furthermore, Rivard discloses that "the memory request generator is coupled between the cache tag block and the cache data store for performing a memory request before an address and instruction information associated with the needed texels reach the cache data store" (Col. 10, lines 48-52 - emphasis added).

Appellant respectfully asserts that a combination of data and instructions in Rivard would result in no need for Rivard's purposefully included "pipeline latency elements," since the combination of data and instructions would arrive together. As a result, there is no suggestion that the modification of the Rivard reference, as argued by the Examiner, is desired. The mere fact that references can be combined or modified does not render the resultant combination obvious unless the prior art also suggests the desirability of the combination. *In re Mills*, 916 F.2d 680, 16 USPQ2d 1430 (Fed. Cir. 1990). Although a prior art device "may be capable of being modified to run the way the apparatus is claimed, there must be a suggestion or motivation in the reference to do so." 916 F.2d at 682, 16 USPQ2d at 1432.).

Additionally, appellant respectfully notes that modifying the Rivard reference, as suggested by the Examiner, would change the principle of operation of the Rivard system, since it would obviate the need for the purposefully included "pipeline latency elements 1025," as noted above. If the proposed modification or combination of the prior art would change the principle of operation of the prior art invention being modified, then the teachings of the references are not sufficient to render the claims *prima facie* obvious. *In re Ratti*, 270 F.2d 810, 123 USPQ 349 (CCPA 1959)

In the Office Action mailed 09/20/2007, the Examiner has failed to specifically respond to appellant's above arguments. Again, appellant respectfully asserts that modifying the Rivard reference, as suggested by the Examiner, would change the principle of operation of the Rivard system, since it would obviate the need for the purposefully included "pipeline latency elements 1025," as noted above.

Furthermore, appellant respectfully asserts that, in the Abstract, Rivard teaches "[a] system for caching texel information in a cache data store, for use in a graphics rendering system which uses interpolative sampling to compute texture color values." Additionally, Rivard teaches that "[t]he system includes a texel memory storing texel information, a graphics application program for using interpolative sampling to compute dynamic texture values, a first cache data storage for a number of the most-recently-retrieved texels, a second cache data storage for a previously-retrieved adjacent line of texels, cache tag blocks for determining whether the texels needed by the graphics accelerator system are cached in either of the first or second cache data stores, and a memory request generator for retrieving texels from texel memory upon indication of a miss by the cache tag blocks."

However, Rivard does not recognize one of the various possible problems solved by appellant, namely to "accommodate the programmability of recent texture and shader modules without being inhibited by the size of associated programs," for example. It was this insight in solving this problem that helped the inventors conceive of the claimed invention which overcomes the drawbacks of the prior art. "Because that insight was contrary to the understandings and expectations of the art, the structure effectuating it would not have been obvious to those skilled in the art." *Schenck v. Nortron Corp.*, 713 F.2d at 785, 218 USPQ at 700 (citations omitted).

In the Office Action mailed 09/20/2007, in paragraph 3 on Page 2, the Examiner has argued the following:

> "In response to [appellant's] argument that the references fail to show certain features of [appellant's] claimed invention, it is noted that the features upon which [appellant] relies (i.e., to accommodate the programmability of recent texture and shader modules without being inhibited by the size of the associated programs) are not recited in the rejected claims(s). Although the claims are interpreted in light of the specification, limitations

from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993)."

Appellant respectfully asserts that Rivard teaches "[a] system for caching texel information in a cache data store, for use in a graphics rendering system which uses interpolative sampling to compute texture color values" (Abstract). Clearly, Rivard's system for caching texel information in a cache data store does not recognize one of the various possible problems solved by appellant, namely to "accommodate the programmability of recent texture and shader modules without being inhibited by the size of associated programs," as disclosed in the Specification, Page 5, lines 13-15, for example. It was the insight in solving the aforementioned problem that helped the inventors conceive of the claimed invention, which overcomes the drawbacks of the prior art. "Because that insight was contrary to the understandings and expectations of the art, the structure effectuating it would not have been obvious to those skilled in the art." *Schenck v. Nortron Corp.*, 713 F.2d at 785, 218 USPQ at 700 (citations omitted).

Additionally, appellant respectfully asserts that the following excerpts from Rivard demonstrate that such reference, in fact, *teaches away* from appellant's claimed invention.

> "If the interface to DRAM 655 is 32-bits wide, the texture mode indicates a 16-bit per pixel texture lookup and the data is conveniently aligned in the texture map, then it is possible to satisfy two lookup requests with a single read request. However, if the data is not aligned, then two read requests are needed." (Col. 6, lines 56-61 - emphasis added)

> "Therefore, graphics accelerator system 635 includes pipeline latency elements 1025 to coordinate arrival of the memory data and of the associated instructions at cache data store and memory data resolver 1030." (Col. 7, lines 3-7 - emphasis added)

> "...the memory request generator is coupled between the cache tag block and the cache data store for performing a memory request before an address and instruction information associated with the needed texels reach the cache data store." (Col. 10, lines 48-52 - emphasis added)

Appellant respectfully asserts that the "pipeline latency elements 1025," as described in Rivard, are introduced specifically "to coordinate arrival of the memory data and of the associated instructions at cache data store and memory data resolver 1030" (emphasis added). Thus, Rivard actually *teaches away* from appellant's claim language by intentionally incorporating the "pipeline latency elements 1025" for the specific purpose of coordinating the arrival of the

instructions and the memory data from separate sources. A *prima facie* case of obviousness may also be rebutted by showing that the art, in any material respect, *teaches away* from the claimed invention. *In re Geisler*, 116 F.3d 1465, 1471, 43 USPQ2d 1362, 1366 (Fed. Cir. 1997).

In the Office Action mailed 09/20/2007, the Examiner has failed to specifically respond to appellant's above arguments. Again, appellant respectfully asserts that Rivard actually *teaches away* from appellant's claim language by intentionally incorporating the "pipeline latency elements 1025" for the specific purpose of coordinating the arrival of the instructions and the memory data from separate sources.

Thus, clearly at least the first element of the *prima facie* case of obviousness has not been met by the Rivard reference.

More importantly, with respect to the third element of the *prima facie* case of obviousness, appellant respectfully asserts that the Rivard reference also fails to meet all of appellant's claim limitations. Specifically, with respect to independent Claims 1, 24, and 27, the Examiner has relied on Figure 6 and Figure 10, in addition to Col. 4, lines 46-57; and Col. 6, line 45-Col. 7, line 10 (excerpted below) from Rivard to make a prior art showing of appellant's claimed "sending an instruction request to video memory, where a texture module... sends the instruction request to the video memory" (see this or similar, but not necessarily identical language in the foregoing independent claims).

> "Graphics application program 670 transfers graphical information from data storage 625 into DRAM 655 for local storage. Graphics accelerator system 635 includes graphics pipeline stages 640 including a texture mapping stage 645 for mapping texture information to the graphics information received from graphics application program 670 and for maintaining the texel information in a texel cache system 650. Texture mapping block 645 sends information via bus 647 to texel cache system 650, and texel cache system 650 sends information via bus 649 back to texture mapping block 645." (Col. 4, lines 46-57 – emphasis added)
>
> "Cache data store 1030 responsively outputs on lines 649 the texture values cached in the read location.
>
> If a miss occurs, then cache tag blocks 1010 and 1015 forward a cache write address to memory request generator 1020. Memory request generator 1020 generates memory requests for all misses (up to four for bi-linear sampling and up to eight for tri-linear sampling), and forwards the requests to DRAM 655 for information retrieval. DRAM 655

returns the memory data on bus 660 to cache data store and memory data resolver 1030, which stores the memory data at the write address. If the interface to DRAM 655 is 32-bits wide, the texture mode indicates a 16-bit per pixel texture lookup and the data is conveniently aligned in the texture map, then it is possible to satisfy two lookup requests with a single read request. However, if the data is not aligned, then two read requests are needed.

Because memory request generator 1020 is between cache tag blocks 1010, 1015 and cache data store 1030, generator 1020 can perform DRAM 655 memory requests before the address and instruction information reach cache data store and memory data resolver 1030. Once memory requests are generated, there is, depending on the DRAM design, a constant latency of about five to ten clock cycles (or possibly more) which includes time for exiting and reentering the graphics pipeline hardware, to effect a page hit. Therefore, graphics accelerator system 635 includes pipeline latency elements 1025 to coordinate arrival of the memory data and of the associated instructions at cache data store and memory data resolver 1030." (Col. 6, line 45-Col. 7, line 10 - emphasis added)

Appellant respectfully asserts that the figures relied on by the Examiner only generally illustrate a computer system and a block diagram detailing the graphics accelerator system showing the pipeline latency elements 1025. In addition, the excerpts relied on by the Examiner merely teach that the "[m]emory request generator 1020 generates memory requests for all misses (up to four for bi-linear sampling and up to eight for tri-linear sampling), and forwards the requests to DRAM 655 for information retrieval" (emphasis added). However, disclosing that a memory request is generated for all misses, as in Rivard, fails to teach "sending an instruction request to video memory, where a texture module... sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

Appellant notes that the Examiner has argued that "the texture mapping stage and the texel cache system together are considered the texture module." Appellant respectfully disagrees and asserts that Rivard merely suggests that the "[g]raphics accelerator system 635 includes graphics pipeline stages 640 including a texture mapping stage 645 for mapping texture information to the graphics information received from graphics application program 670 and for maintaining the texel information in a texel cache system 650" where the "[t]exture mapping block 645 sends information via bus 647 to texel cache system 650, and texel cache system 650 sends information via bus 649 back to texture mapping block 645" (Col. 4, lines 49-57 – emphasis added). Clearly, the texture mapping stage **is separate from** the texture cache system, as clearly disclosed in Rivard, and thus fails to suggest "sending an instruction request to video memory, where a

texture module... sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

In addition, the Examiner has argued that "memory requests/read requests for all misses that are forwarded to DRAM are instruction requests based on which DRAM sends back information... [where] the DRAM sends memory data based on the memory requests/read requests from the texture module." Further, the Examiner has argued that "[m]emory requests/read requests act as an instruction to DRAM, which performs a particular function based on the request."

Appellant respectfully disagrees, and asserts that Rivard merely discloses that the "[t]exture mapping block 645 sends information via bus 647 to texel cache system 650, and texel cache system 650 sends information via bus 649 back to texture mapping block 645" (Col. 4, lines 54-57). Additionally, Rivard discloses that "[t]exel sample address computation block 1005 forwards the higher resolution sample points A-D to cache tag block 1010 and forwards the lower resolution sample points E-H to cache tag block 1015" and "[i]f a miss occurs, then cache tag blocks 1010 and 1015 forward a cache write address to memory request generator 1020" which "generates memory requests for all misses... and forwards the requests to DRAM 655 for information retrieval" (Col. 6, lines 33-36, and lines 48-53 – emphasis added).

However, Rivard's disclosure that the memory request generator 1020, which is included in the texel cache system (see Figure 10), generates memory requests for all misses and forwards the requests to DRAM for information retrieval, simply fails to even suggest "sending an instruction request to video memory, where a texture module... sends the instruction request to the video memory" (emphasis added), as claimed by appellant. Moreover, the texel sample address computation block (included in texture mapping block 645) which forwards sample points to the cache tag blocks 1010 and 1015 (included in texel cache system 650), as disclosed in Rivard, fails to meet "sending an instruction request to video memory, where a texture module... sends the instruction request to the video memory" (emphasis added), as claimed by appellant. Furthermore, the memory request to the DRAM, as in Rivard, fails to suggest "an instruction request," in the manner as claimed by appellant.

In addition, appellant respectfully asserts that the following excerpts from Rivard further demonstrate that the Rivard fails to disclose appellant's claimed "sending an instruction request to video memory, where a texture module... sends the instruction request to the video memory" (see this or similar, but not necessarily identical language in the foregoing independent claims).

> "If the interface to DRAM 655 is 32-bits wide, the texture mode indicates a 16-bit per pixel texture lookup and the data is conveniently aligned in the texture map, then it is possible to satisfy two lookup requests with a single read request. However, if the data is not aligned, then two read requests are needed." (Col. 6, lines 56-61 - emphasis added)
>
> "Therefore, graphics accelerator system 635 includes pipeline latency elements 1025 to coordinate arrival of the memory data and of the associated instructions at cache data store and memory data resolver 1030." (Col. 7, lines 3-7 - emphasis added)
>
> "...the memory request generator is coupled between the cache tag block and the cache data store for performing a memory request before an address and instruction information associated with the needed texels reach the cache data store." (Col. 10, lines 48-52 - emphasis added)

First, appellant respectfully points out that Rivard merely teaches that "the memory request generator is coupled between the cache tag block and the cache data store for performing a memory request before an address and instruction information associated with the needed texels reach the cache data store" (emphasis added). Clearly, coordinating the arrival of "memory data", where "the data is conveniently aligned in the texture map" (emphasis added), as in Rivard (see excerpts above), fails to teach "sending an instruction request" (emphasis added), as claimed by appellant.

Second, appellant respectfully asserts that Rivard simply discloses that a memory request is performed before an address and instruction information associated with the needed texels reach the cache data store (see excerpts above). Thus, the memory data in Rivard simply relates to texel data, which clearly fails to suggest "sending an instruction request to video memory, where a texture module... sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

Furthermore, the Examiner has argued that "the definition of instruction provided by dictionary.com, which defines instructions as a command given to a computer to carry out a

particular operation and can contain data to be used in the operation; here DRAM sends memory data based on the memory requests/read requests from the texture module." Further, the Examiner has argued that "[m]emory requests/read requests act as an instruction to DRAM, which performs a particular function based on the request."

Appellant respectfully disagrees with the Examiner's arguments and asserts that the dictionary.com reference provided by the Examiner merely defines an instruction as "a command given to a computer to carry out a particular operation." Appellant respectfully asserts that Rivard merely discloses that the "cache tag blocks 1010 and 1015 forward a cache write address to memory request generator 1020" which "generates memory requests for all misses... and forwards the requests to DRAM 655 for information retrieval" (Col. 6, lines 33-36, and lines 48-53 – emphasis added). However, the disclosure of the memory request generator generating a memory request and forwarding the request to DRAM for information retrieval, as in Rivard, simply fails to suggest "sending an instruction request to video memory, where a texture module... sends the instruction request to the video memory" (emphasis added), as claimed by appellant. Clearly, generating and forwarding a memory request to DRAM, as in Rivard, fails to suggest "sending an instruction request," particularly where "instructions [are received]...in response to the instruction request" (emphasis added), in the context as claimed by appellant.

In the Office Action mailed 09/20/2007, on Page 2, line 19 to Page 3, line 19; and Page 4, lines 3-7, the Examiner has argued the following:

> "However, the examiner interprets that Rivard (Fig. 6, Fig. 10, col. 4 lines 46-57, col. 6 lines 45-67, col. 7 lines 1-10) teaches sending an instruction request to video memory, where a texture module in a graphics pipeline sends the instruction request to the video memory (a texture mapping stage sends information to texel cache system; the texture mapping stage and the texel cache system together are considered as texture module, so the information is passed between the components of the texture module; texel cache system comprising the memory request generator, pipeline latency elements, and cache data store and memory data resolver forwards the memory requests/read requests to DRAM for information retrieval. [T]he examiner interprets that Rivard discloses sending an instruction request to video memory by generating memory request for all misses, and forwards the requests to DRAM for information retrieval (Fig. 6, Fig. 10, col. 4 lines 46-57, col. 6 lines 45-67, col. 7 lines 1-10). It is interpreted that an instruction request could be considered as merely a request because it is basically a request that requires DRAM to perform necessary operation. The examiner also interprets that instructions stored in DRAM that are requested are basically stored as data. It is further interpreted that

> memory requests/read requests act as an instruction to DRAM based on which DRAM performs a particular function. Memory requests/read requests for all misses that are forwarded to DRAM are instruction requests based on which DRAM sends back information; the examiner relies on the definition of instruction provided by dictionary.com, which defines instructions as a command given to a computer to carry out a particular operation and can contain data to be used in the operation; here DRAM sends memory based on the memory requests/read requests from the texture module." (Page 2, line 19 to Page 3, line 19)
>
> "In response to [appellant's] arguments against the references individually, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986)." (Page 4, lines 3-7)

First, appellant notes that the Examiner's arguments on Page 4, line 8 to Page 5, line 8 in the Office Action mailed 09/20/2007 are substantially similar to the arguments made by the Examiner on Page 2, line 19 to Page 3, line 19 excerpted above. Second, appellant has clearly responded to such arguments above regarding how the excerpts repeatedly relied upon by the Examiner fail to meet or suggest appellant's claimed "sending an instruction request to video memory, where a texture module... sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

Further, appellant respectfully disagrees with the Examiner's arguments and asserts that the figures and excerpts from Rivard relied upon by the Examiner merely disclose that "the texture mode indicates a 16-bit per pixel texture lookup" in addition to "satisfy[ing] two lookup requests with a single read request" (Col. 6, lines 56-61 – emphasis added). Additionally, the excerpts from Rivard teach that "pipeline latency elements 1025... coordinate arrival of the memory data and of the associated instructions" (Col. 7, lines 3-7 – emphasis added) such that "the memory request generator... perform[s] a memory request before an address and instruction information associated with the needed texels reach the cache data store" (Col. 10, lines 48-52 – emphasis added). Clearly, the texture mode indicating a texture lookup, in addition to the memory request generator performing a memory request, as in Rivard, simply fails to support the Examiner's allegation that "a texture module... sends the instruction request to the video memory" (emphasis added), and especially does not teach "sending an instruction request to video memory, where a texture module... sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

In addition, Rivard's disclosure that the "cache tag blocks 1010 and 1015 forward a cache write address to memory request generator 1020," which "generates memory requests for all misses ... and forwards the requests to DRAM 655 for information retrieval" (Col. 6, lines 48-53 – emphasis added). However, the disclosure of the memory request generator generating a memory request and forwarding the request to DRAM for information retrieval, as in Rivard, fails to support the Examiner's allegation "that Rivard discloses sending an instruction request to video memory by generating memory request for all misses, and forwards the requests to DRAM for information retrieval" (emphasis added). Clearly, generating a memory request and forwarding the request to DRAM for information retrieval, as in Rivard, simply fails to specifically teach "sending an instruction request to video memory, where a texture module... sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

Still yet, appellant respectfully asserts that the excerpts from Rivard relied upon by the Examiner disclose that the "[g]raphics application program 670 transfers graphical information from data storage 625 into DRAM 655 for local storage" (Col. 4, lines 46-48 – emphasis added). Additionally, the excerpts disclose that "[i]f a hit occurs, then cache tag blocks 1010 and 1015 forward a cache read address through memory request generator 1020 and through pipeline latency elements 1025 to cache data store 1030" such that "[c]ache data store 1030 responsively outputs on lines 649 the texture values cached in the read location" (Col. 6, lines 42-47 – emphasis added). Clearly, Rivard is disclosing the transfer of graphical information into DRAM and outputting cached texture values, which simply fails to support the Examiner's allegation that "instructions stored in DRAM that are requested are basically stored as data" (emphasis added). Therefore, the disclosure of transferring graphical information into DRAM and outputting cached texture values, as in Rivard, simply fails to suggest "sending an instruction request to video memory, where a texture module... sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

Furthermore, appellant again asserts that the dictionary.com reference provided by the Examiner merely defines an instruction as "a command given to a computer to carry out a particular operation." In addition, the excerpts from Rivard disclose that the "memory request generator 1020 generates memory requests for all misses... and forwards the requests to DRAM 655 for

information retrieval" (Col. 6, lines 50-53 – emphasis added). However, generating and forwarding memory requests to DRAM for information retrieval, as in Rivard, fails to support the Examiner's allegation that "[m]emory requests/read requests for all misses that are forwarded to DRAM are instruction requests based on which DRAM sends back information" (emphasis added). Clearly, generating and forwarding memory requests to DRAM for information retrieval, as in Rivard, simply fails to specifically teach "sending an instruction request to video memory, where a texture module... sends the instruction request to the video memory" (emphasis added), as claimed by appellant. Applicant emphasizes that the memory request of Rivard fails to meet appellant's claimed "instruction request," as claimed by appellant.

Additionally, with respect to independent Claims 1, 24, and 27, the Examiner has relied on Figure 6 and Figure 10; Col. 4, lines 46-57; Col. 6, lines 45-67; and Col. 7, lines 1-10 from Rivard (reproduced above) to make a prior art showing of appellant's claimed "receiving instructions from the video memory in response to the instruction request utilizing the texture module in the graphics pipeline" (see this or similar, but not necessarily identical language in the foregoing independent claims).

Specifically, the Examiner has argued that "DRAM returns memory data to cache data store and memory data resolver component of texel cache system, which further sends this information to the texture mapping stage; [where] the texture mapping stage and the texel cache system together are considered the cache module, so the information/memory data is passed between the components of the texture module."

Appellant respectfully disagrees, and points out that in the second paragraph on Page 7 of the Office Action mailed 03/20/2007, the Examiner has stated that "Rivard does not explicitly teach to combine texture mapping stage and texel cache system to form a texture module," and thus, Rivard simply fails to suggest "receiving instructions from the video memory in response to the instruction request utilizing the texture module in the graphics pipeline" (emphasis added), as claimed by appellant.

In addition, the excerpts from Rivard relied upon by the Examiner merely teach that "DRAM 655 returns the **memory data** on bus 660 to cache data store and memory data resolver 1030"

(emphasis added). However, merely returning memory data to a cache data store, as in Rivard, simply fails to teach "receiving instructions from the video memory in response to the instruction request utilizing the texture module in the graphics pipeline" (emphasis added), as claimed by appellant.

Furthermore, appellant respectfully asserts that Rivard teaches that "[t]exture mapping block 645 sends information via bus 647 to texel cache system 650, and texel cache system 650 sends information via bus 649 back to texture mapping block 645" (Col. 4, lines 54-57) Additionally, Rivard discloses that "[t]exel sample address computation block 1005 forwards the higher resolution sample points A-D to cache tag block 1010 and forwards the lower resolution sample points E-H to cache tag block 1015" and "[i]f a miss occurs, then cache tag blocks 1010 and 1015 forward a cache write address to memory request generator 1020" which "generates memory requests for all misses... and forwards the requests to DRAM 655 for information retrieval" (Col. 6, lines 33-36, and lines 48-53 – emphasis added). Further, Rivard discloses that "DRAM 655 returns the memory data on bus 660 to cache data store and memory data resolver 1030, which stores the memory data at the write address" (Col. 6, lines 53-56 – emphasis added).

However, the disclosure of a memory request generator that generates memory requests for all misses and forwards the requests to DRAM for information retrieval, and that the DRAM then returns the memory data which is stored at the write address, as in Rivard, simply fails to even suggest "receiving instructions from the video memory in response to the instruction request utilizing the texture module in the graphics pipeline" (emphasis added), as claimed by appellant. Clearly, returning memory data which is stored at a write address after a cache miss, as in Rivard, fails to meet "receiving instructions from the video memory in response to the instruction request" (emphasis added), as claimed by appellant. Appellant emphasizes that the memory data from the DRAM, as disclosed in Rivard, fails to teach or suggest "instructions," in the manner as claimed by appellant.

Furthermore, in the Office Action mailed 09/20/2007, on Page 7, lines 3-7; Page 5, lines 9-17; Page 6, line 10 to Page 7, line 4; and Page 8, lines 4-8, the Examiner has argued the following:

> "In response to [appellant's] arguments against the references individually, one cannot show nonobviousness by attacking references individually where the rejections are based

on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986)." (Page 7, lines 3-7)

"Rivard further teaches receiving instructions from the video memory in response to the instruction request utilizing the texture module in the graphics pipeline (it is interpreted that instructions stored in DRAM that are requested are basically stored as data, and Rivard teaches to send back data in some form in response to the memory request generated as a result of a miss; DRAM returns memory data to cache data store and memory data resolver component of texel cache system, which further sends this information to the texture mapping stage; the texture mapping stage and the texel cache system together are considered as texture module, so the information/memory data is passed between the components of the texture module)." (Page 5, lines 9-17)

"The examiner further interprets that Rivard also teaches the arrival of memory data and associated instructions at the cache data store and memory data resolver component of the text cache system is coordinated by pipeline latency elements (col. 7 lines 4-7; memory data has associated instructions are returned by DRAM, though the data has it's associated instructions). Therefore, the examiner brings in the Wang reference that suggests a graphics subunit (texture module) in a graphics hardware system that supplies data and control signals to local frame buffer memory for executing a series of display instructions (Fig. 1, col. 5 lines 38-67; the computer memory includes the local frame buffer memory, which corresponds to the DRAM; based on the control signals send by the graphics hardware system, the frame buffer returns the polygon display instructions that includes the texture data, so that the graphics subunit executes the display instructions using this data). Therefore, it would have been obvious to one of ordinary skill in art at the time of present invention to have the memory return instructions as taught by Wang to the texture module of Rivard because these instructions are needed to render the graphics primitives to be displayed on the display device (col. 5 lines 43-45 and lines 63-67)." (Page 6, line 10 to Page 7, line 4)

"However, the examiner interprets that Rivard teaches to send a request to DRAM via a texture module, and based on this request, DRAM sends back data and it's associated information to the texture module (see the arguments above for details). Nonetheless, Rivard does not explicitly state that the data and it's associated instructions are returned by DRAM, though the data has it's associated functions." (Page 8, lines 4-8)

First, appellant notes that the Examiner's arguments on Page 7, lines 5-21; and Page 8, lines 9-20 in the Office Action mailed 09/20/2007 are substantially similar to the arguments made by the Examiner on Page 6, line 10 to Page 7, line 4 excerpted above. Further, on Page 7, line 21 the Examiner has additionally citied "*In re Lockhart*, 90 USPQ (CCPA 1951)." Second, appellant has clearly responded to such arguments above regarding how the excerpts repeatedly relied upon by the Examiner fail to meet or suggest appellant's claimed "receiving instructions from

the video memory in response to the instruction request utilizing the texture module in the graphics pipeline" (emphasis added), as claimed by appellant.

Further, appellant respectfully asserts that the excerpts from Rivard relied upon by the Examiner merely teach that "[m]emory request generator 1020 generates memory requests for all misses... and forwards the requests to DRAM 655 for information retrieval" such that "DRAM 655 returns the memory data on bus 660 to cache data store and memory data resolver 1030" (Col. 6, lines 50-55 – emphasis added). However, Rivard's disclosure that memory requests are forwarded to DRAM for information retrieval, where the DRAM returns the memory data, fails to support the Examiner's allegation that "Rivard further teaches receiving instructions from the video memory in response to the instruction request utilizing the texture module in the graphics pipeline" (emphasis added). Clearly, forwarding memory requests to DRAM for information retrieval, as in Rivard, simply fails to specifically teach "receiving instructions from the video memory in response to the instruction request utilizing the texture module in the graphics pipeline" (emphasis added), as claimed by appellant.

In addition, appellant respectfully asserts that the excerpts from Rivard relied upon by the Examiner teach that "[b]ecause memory request generator 1020 is between cache tag blocks 1010, 1015 and cache data store 1030, generator 1020 can perform DRAM 655 memory requests before the address and instruction information reach cache data store and memory data resolver 1030" (Col. 6, lines 62-66 – emphasis added). Furthermore, Wang teaches "a 3D graphics subunit 109 for executing a series of display instructions found within a display list stored in computer memory" and that "[m]any of the polygon display instructions include texture data to be displayed within the polygon" (Col. 5, lines 40-48 – emphasis added). However, the mere disclosure of a memory request generator 1020 performing DRAM memory requests before the instruction information reaches the cache data store, as in Rivard, in addition to the disclosure of executing display instructions that may include texture data, as in Wang, simply fails to suggest "receiving instructions from the video memory in response to the instruction request utilizing the texture module in the graphics pipeline" (emphasis added), as claimed by appellant. Clearly, performing memory requests before instruction information reaches the cache data store, as in Rivard, in addition to the disclosure of executing display instructions that may include texture data, as in Wang, simply fails to even suggest an "instruction request," much less "receiving

instructions from the video memory in response to the instruction request utilizing the texture module in the graphics pipeline" (emphasis added), as claimed by appellant.

Further, appellant respectfully asserts that Rivard's disclosure of a "graphics accelerator system 635 [that] includes pipeline latency elements 1025 to coordinate arrival of the memory data and of the associated instructions at cache data store and memory data resolver 1030" (Col. 7, lines 4-7) clearly *teaches away* from the Examiner's allegation that it would be obvious to "have the memory return instructions as taught by Wang to the texture module of Rivard because these instructions are needed to render the graphics primitives to be displayed on the display device," as noted by the Examiner, since in Rivard the "pipeline latency elements… coordinate arrival of the memory data and… the associated instructions" (emphasis added). Clearly, as argued above, it would not be obvious from the teachings of Rivard and Wang to "receiv[e] instructions from the video memory in response to the instruction request utilizing the texture module in the graphics pipeline" (emphasis added), as claimed by appellant.

Still yet, appellant agrees with the Examiner's statement that "Rivard does not explicitly state that the data and it's associated instructions are returned by DRAM." For example, appellant asserts that the figures and excerpts from Rivard relied upon by the Examiner teach that texel cache system 650 "includes cache tags 1010, cache tags 1015, a memory request generator 1020, pipeline latency elements 1025 and cache data store and memory data resolver 1030" (see Figure 10; Col. 6, lines 22-26 – emphasis added), where the "[m]emory request generator 1020 generates memory requests for all misses… and forwards the requests to DRAM 655 for information retrieval" (Col. 6, lines 50-53 – emphasis added). Clearly, a texel cache system 650 including a memory request generator 1020 that generates memory requests that are forwarded to DRAM 655 for information retrieval, as in Rivard, simply fails to suggest "receiving instructions from the video memory in response to the instruction request utilizing the texture module in the graphics pipeline" (emphasis added), as claimed by appellant.

In addition, the Examiner has admitted that "Rivard does not explicitly teach to combine texture mapping stage and texel cache system to form a texture module," and has argued that "it would have been obvious to one of ordinary skill in the art at the time of present invention to combine texture mapping stage and texel cache system of Rivard to work together as a texture module."

Appellant disagrees and respectfully asserts that it would not have been obvious to combine the texture mapping stage and texel cache system in Rivard, as suggested by the Examiner. Specifically, Rivard teaches the use of "pipeline latency elements 1025" which "coordinate arrival of the memory data and of the associated instructions at cache data store and memory data resolver 1030" (Col. 7, lines 3-7 - emphasis added). However, Rivard actually *teaches away* from the Examiner's above allegation, in addition to appellant's claim language, by intentionally incorporating the pipeline latency elements to coordinate arrival of the memory data from separate sources. Therefore, for the reasons argued above, it would not be obvious to "combine texture mapping stage and texel cache system," as argued by the Examiner. Thus, it would not have been obvious to "receiv[e] instructions from the video memory in response to the instruction request utilizing the texture module..." (emphasis added), as appellant claims.

Appellant thus formally requests a specific showing of the subject matter in ALL of the claims in any future action. Note excerpt from MPEP below.

> "If the appellant traverses such an [Official Notice] assertion the examiner should cite a reference in support of his or her position." See MPEP 2144.03.

Still yet, the Examiner has admitted that "Rivard does not explicitly teach that the memory returns instructions along with data, in response to [the] instruction request from the texture module," but has argued that "Wang teaches a graphics subunit (texture module) in a graphics hardware system that supplies data and control signals to local frame buffer memory for executing a series of display instructions (Fig. 1, col. 5 lines 38-67...)."

Appellant respectfully disagrees. The excerpt from Wang relied on by the Examiner merely discloses a "graphics subunit 109 for executing a series of display instructions found within a display list stored in computer memory." However, only generally disclosing that a graphics subunit executes instructions stored in computer memory, as in Wang, fails to specifically meet appellant's claimed "receiving instructions from the video memory in response to the instruction request utilizing the texture module..." (emphasis added), as appellant claims. In fact, appellant notes that Wang only discloses that the "graphics subunit 109 includ[es] a texture engine 10,

[and] a polygon engine 12," and that the "texture engine 10 is responsible for retrieving the texture map data," whereas the "polygon engine 12…performs well known polygon rendering functions" (Col. 6, lines 3-49). Thus, Wang clearly does not disclose "receiving instructions…utilizing the texture module…" (emphasis added), as claimed.

Furthermore, in the Office Action mailed 09/20/2007, on Page 5, line 18 to Page 6, line 9, the Examiner has argued the following:

> "Although Rivard discloses the limitations as stated above, Rivard does not explicitly teach to combine texture mapping stage and texel cache system to form a texture module. However, it would have been obvious to one of ordinary skill in art at the time of present invention to combine texture mapping stage and texel cache system of Rivard to work together as a texture module. The unity of diversity of parts would depend more upon the choice of the manufacturer, and the convenience and availability of the machines and tools necessary to construct the texture module, than on any inventive concept. One of ordinary skill in art, furthermore, would have expected [appellant's] invention to perform equally well with Rivard's reference that teaches to send and receive information/instruction from the texture mapping stage and texel cache system to the DRAM because using this components together will also result in sending and receiving instructions to and from DRAM. Therefore, it would have been obvious to one of ordinary skill in art at the time of present invention to modify Rivard to obtain the invention as specified in the claim." (Page 5, line 18 to Page 6, line 9)

First, appellant respectfully asserts that the Examiner's arguments on Page 7, lines 8-21 of the Office Action mailed 09/20/2007 are substantially similar to the arguments made by the Examiner on Page 5, line 18 to Page 6, line 9 excerpted above. Further, on Page 7, line 21 the Examiner has additionally citied "*In re Lockhart*, 90 USPQ (CCPA 1951)."

Further, appellant respectfully disagrees with the Examiner's arguments and asserts that it would not be obvious to "combine texture mapping stage and texel cache system of Rivard to work together as a texture module," as alleged by the Examiner. For example, Rivard teaches that a "[g]raphics accelerator system 635 includes graphics pipeline stages 640 including a texture mapping stage 645 for mapping texture information to the graphics information received from graphics application program 670 and for maintaining the texel information in a texel cache system 650" such that "[t]exture mapping block 645 sends information via bus 647 to texel cache system 650, and texel cache system 650 sends information via bus 649 back to texture mapping block 645" (Col. 4, lines 49-57 – emphasis added). Clearly, Rivard teaches and suggests the use

of graphic pipeline stages, such as a texture mapping block 645 and texel cache system 650, which simply fails to support the Examiner's allegation that it would be obvious to "combine texture mapping stage and texel cache system of Rivard to work together as a texture module." Therefore, Rivard's teachings of a separate texture mapping block 645 and texel cache system 650 simply fails to suggest "receiving instructions...utilizing the texture module..." (emphasis added), as claimed.

Again, appellant respectfully asserts that at least the first and third elements of the *prima facie* case of obviousness have not been met, since it would be *unobvious* to combine the references, as noted above, and the prior art excerpts, as relied upon by the Examiner, fail to teach or suggest all of the claim limitations, as noted above.

Group #2: Claims 25 and 26

With respect to independent Claims 25 and 26, the Examiner has relied on Figure 6 and Figure 10, in addition to Col. 4, lines 46-57; and Col. 6, line 45-Col. 7, line 10 (excerpted below) from Rivard to make a prior art showing of appellant's claimed "sending an instruction request to video memory, where a texture module sends the instruction request to the video memory" (see this or similar, but not necessarily identical language in the foregoing independent claims).

> "Graphics application program 670 transfers graphical information from data storage 625 into DRAM 655 for local storage. Graphics accelerator system 635 includes graphics pipeline stages 640 including a texture mapping stage 645 for mapping texture information to the graphics information received from graphics application program 670 and for maintaining the texel information in a texel cache system 650. Texture mapping block 645 sends information via bus 647 to texel cache system 650, and texel cache system 650 sends information via bus 649 back to texture mapping block 645." (Col. 4, lines 46-57 – emphasis added)
>
> "Cache data store 1030 responsively outputs on lines 649 the texture values cached in the read location.
>
> If a miss occurs, then cache tag blocks 1010 and 1015 forward a cache write address to memory request generator 1020. Memory request generator 1020 generates memory requests for all misses (up to four for bi-linear sampling and up to eight for tri-linear sampling), and forwards the requests to DRAM 655 for information retrieval. DRAM 655 returns the memory data on bus 660 to cache data store and memory data resolver 1030, which stores the memory data at the write address. If the interface to DRAM 655 is 32-bits wide, the texture mode indicates a 16-bit per pixel texture lookup and the data is conveniently aligned in

> the texture map, then it is possible to satisfy two lookup requests with a single read request. However, if the data is not aligned, then two read requests are needed.
>
> Because memory request generator 1020 is between cache tag blocks 1010, 1015 and cache data store 1030, generator 1020 can perform DRAM 655 memory requests before the address and instruction information reach cache data store and memory data resolver 1030. Once memory requests are generated, there is, depending on the DRAM design, a constant latency of about five to ten clock cycles (or possibly more) which includes time for exiting and reentering the graphics pipeline hardware, to effect a page hit. Therefore, graphics accelerator system 635 includes pipeline latency elements 1025 to coordinate arrival of the memory data and of the associated instructions at cache data store and memory data resolver 1030." (Col. 6, line 45-Col. 7, line 10 - emphasis added)

Appellant respectfully asserts that the figures relied on by the Examiner only generally illustrate a computer system and a block diagram detailing the graphics accelerator system showing the pipeline latency elements 1025. In addition, the excerpts relied on by the Examiner merely teach that the "[m]emory request generator 1020 generates memory requests for all misses (up to four for bi-linear sampling and up to eight for tri-linear sampling), and forwards the requests to DRAM 655 for information retrieval" (emphasis added). However, disclosing that a memory request is generated for all misses, as in Rivard, fails to teach "sending an instruction request to video memory, where a texture module sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

Appellant notes that the Examiner has argued that "the texture mapping stage and the texel cache system together are considered the texture module." Appellant respectfully disagrees and asserts that Rivard merely suggests that the "[g]raphics accelerator system 635 includes graphics pipeline stages 640 including a texture mapping stage 645 for mapping texture information to the graphics information received from graphics application program 670 and for maintaining the texel information in a texel cache system 650" where the "[t]exture mapping block 645 sends information via bus 647 to texel cache system 650, and texel cache system 650 sends information via bus 649 back to texture mapping block 645" (Col. 4, lines 49-57 – emphasis added). Clearly, the texture mapping stage **is separate from** the texture cache system, as clearly disclosed in Rivard, and thus fails to suggest "sending an instruction request to video memory, where a texture module sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

In addition, the Examiner has argued that "memory requests/read requests for all misses that are forwarded to DRAM are instruction requests based on which DRAM sends back information... [where] the DRAM sends memory data based on the memory requests/read requests from the texture module." Further, the Examiner has argued that "[m]emory requests/read requests act as an instruction to DRAM, which performs a particular function based on the request."

Appellant respectfully disagrees, and asserts that Rivard merely discloses that the "[t]exture mapping block 645 sends information via bus 647 to texel cache system 650, and texel cache system 650 sends information via bus 649 back to texture mapping block 645" (Col. 4, lines 54-57). Additionally, Rivard discloses that "[t]exel sample address computation block 1005 forwards the higher resolution sample points A-D to cache tag block 1010 and forwards the lower resolution sample points E-H to cache tag block 1015" and "[i]f a miss occurs, then cache tag blocks 1010 and 1015 forward a cache write address to memory request generator 1020" which "generates memory requests for all misses... and forwards the requests to DRAM 655 for information retrieval" (Col. 6, lines 33-36, and lines 48-53 – emphasis added).

However, Rivard's disclosure that the memory request generator 1020, which is included in the texel cache system (see Figure 10), generates memory requests for all misses and forwards the requests to DRAM for information retrieval, simply fails to even suggest "sending an instruction request to video memory, where a texture module sends the instruction request to the video memory" (emphasis added), as claimed by appellant. Moreover, the texel sample address computation block (included in texture mapping block 645) which forwards sample points to the cache tag blocks 1010 and 1015 (included in texel cache system 650), as disclosed in Rivard, fails to meet "sending an instruction request to video memory, where a texture module sends the instruction request to the video memory" (emphasis added), as claimed by appellant. Furthermore, the memory request to the DRAM, as in Rivard, fails to suggest "an instruction request," in the manner as claimed by appellant.

In addition, appellant respectfully asserts that the following excerpts from Rivard further demonstrate that the Rivard fails to disclose appellant's claimed "sending an instruction request

to video memory, where a texture module sends the instruction request to the video memory" (see this or similar, but not necessarily identical language in the foregoing independent claims).

> "If the interface to DRAM 655 is 32-bits wide, the texture mode indicates a 16-bit per pixel texture lookup and the data is conveniently aligned in the texture map, then it is possible to satisfy two lookup requests with a single read request. However, if the data is not aligned, then two read requests are needed." (Col. 6, lines 56-61 - emphasis added)
>
> "Therefore, graphics accelerator system 635 includes pipeline latency elements 1025 to coordinate arrival of the memory data and of the associated instructions at cache data store and memory data resolver 1030." (Col. 7, lines 3-7 - emphasis added)
>
> "...the memory request generator is coupled between the cache tag block and the cache data store for performing a memory request before an address and instruction information associated with the needed texels reach the cache data store." (Col. 10, lines 48-52 - emphasis added)

First, appellant respectfully points out that Rivard merely teaches that "the memory request generator is coupled between the cache tag block and the cache data store for performing a memory request before an address and instruction information associated with the needed texels reach the cache data store" (emphasis added). Clearly, coordinating the arrival of "memory data", where "the data is conveniently aligned in the texture map" (emphasis added), as in Rivard (see excerpts above), fails to teach "sending an instruction request" (emphasis added), as claimed by appellant.

Second, appellant respectfully asserts that Rivard simply discloses that a memory request is performed before an address and instruction information associated with the needed texels reach the cache data store (see excerpts above). Thus, the memory data in Rivard simply relates to texel data, which clearly fails to suggest "sending an instruction request to video memory, where a texture module sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

Furthermore, the Examiner has argued that "the definition of instruction provided by dictionary.com, which defines instructions as a command given to a computer to carry out a particular operation and can contain data to be used in the operation: here DRAM sends memory data based on the memory requests/read requests from the texture module." Further, the

Examiner has argued that "[m]emory requests/read requests act as an instruction to DRAM, which performs a particular function based on the request."

Appellant respectfully disagrees with the Examiner's arguments and asserts that the dictionary.com reference provided by the Examiner merely defines an instruction as "a command given to a computer to carry out a particular operation." Appellant respectfully asserts that Rivard merely discloses that the "cache tag blocks 1010 and 1015 forward a cache write address to memory request generator 1020" which "generates memory requests for all misses... and forwards the requests to DRAM 655 for information retrieval" (Col. 6, lines 33-36, and lines 48-53 – emphasis added). However, the disclosure of the memory request generator generating a memory request and forwarding the request to DRAM for information retrieval, as in Rivard, simply fails to suggest "sending an instruction request to video memory, where a texture module sends the instruction request to the video memory" (emphasis added), as claimed by appellant. Clearly, generating and forwarding a memory request to DRAM, as in Rivard, fails to suggest "sending an instruction request," particularly where "instructions [are received]...in response to the instruction request" (emphasis added), in the context as claimed by appellant.

In the Office Action mailed 09/20/2007, on Page 2, line 19 to Page 3, line 19; and Page 4, lines 3-7, the Examiner has argued the following:

> "However, the examiner interprets that Rivard (Fig. 6, Fig. 10, col. 4 lines 46-57, col. 6 lines 45-67, col. 7 lines 1-10) teaches sending an instruction request to video memory, where a texture module in a graphics pipeline sends the instruction request to the video memory (a texture mapping stage sends information to texel cache system; the texture mapping stage and the texel cache system together are considered as texture module, so the information is passed between the components of the texture module; texel cache system comprising the memory request generator, pipeline latency elements, and cache data store and memory data resolver forwards the memory requests/read requests to DRAM for information retrieval. [T]he examiner interprets that Rivard discloses sending an instruction request to video memory by generating memory request for all misses, and forwards the requests to DRAM for information retrieval (Fig. 6, Fig. 10, col. 4 lines 46-57, col. 6 lines 45-67, col. 7 lines 1-10). It is interpreted that an instruction request could be considered as merely a request because it is basically a request that requires DRAM to perform necessary operation. The examiner also interprets that instructions stored in DRAM that are requested are basically stored as data. It is further interpreted that memory requests/read requests act as an instruction to DRAM based on which DRAM performs a particular function. Memory requests/read requests for all misses that are forwarded to DRAM are instruction requests based on which DRAM sends back

information; the examiner relies on the definition of instruction provided by dictionary.com, which defines instructions as a command given to a computer to carry out a particular operation and can contain data to be used in the operation; here DRAM sends memory based on the memory requests/read requests from the texture module." (Page 2, line 19 to Page 3, line 19)

"In response to [appellant's] arguments against the references individually, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986)." (Page 4, lines 3-7)

First, appellant notes that the Examiner's arguments on Page 4, line 8 to Page 5, line 8 in the Office Action mailed 09/20/2007 are substantially similar to the arguments made by the Examiner on Page 2, line 19 to Page 3, line 19 excerpted above. Second, appellant has clearly responded to such arguments above regarding how the excerpts repeatedly relied upon by the Examiner fail to meet or suggest appellant's claimed "sending an instruction request to video memory, where a texture module sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

Further, appellant respectfully disagrees with the Examiner's arguments and asserts that the figures and excerpts from Rivard relied upon by the Examiner merely disclose that "the texture mode indicates a 16-bit per pixel texture lookup" in addition to "satisfy[ing] two lookup requests with a single read request" (Col. 6, lines 56-61 – emphasis added). Additionally, the excerpts from Rivard teach that "pipeline latency elements 1025... coordinate arrival of the memory data and of the associated instructions" (Col. 7, lines 3-7 – emphasis added) such that "the memory request generator... perform[s] a memory request before an address and instruction information associated with the needed texels reach the cache data store" (Col. 10, lines 48-52 – emphasis added). Clearly, the texture mode indicating a texture lookup, in addition to the memory request generator performing a memory request, as in Rivard, simply fails to support the Examiner's allegation that "a texture module in a graphics pipeline sends the instruction request to the video memory" (emphasis added), and especially does not teach "sending an instruction request to video memory, where a texture module sends the instruction request to the video memory" (emphasis added), as claimed by appellant..

In addition, Rivard's disclosure that the "cache tag blocks 1010 and 1015 forward a cache write address to memory request generator 1020," which "generates memory requests for all misses... and forwards the requests to DRAM 655 for information retrieval" (Col. 6, lines 48-53 – emphasis added). However, the disclosure of the memory request generator generating a memory request and forwarding the request to DRAM for information retrieval, as in Rivard, fails to support the Examiner's allegation "that Rivard discloses sending an instruction request to video memory by generating memory request for all misses, and forwards the requests to DRAM for information retrieval" (emphasis added). Clearly, generating a memory request and forwarding the request to DRAM for information retrieval, as in Rivard, simply fails to even suggest specifically teach "sending an instruction request to video memory, where a texture module sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

Still yet, appellant respectfully asserts that the excerpts from Rivard relied upon by the Examiner disclose that the "[g]raphics application program 670 transfers graphical information from data storage 625 into DRAM 655 for local storage" (Col. 4, lines 46-48 – emphasis added). Additionally, the excerpts disclose that "[i]f a hit occurs, then cache tag blocks 1010 and 1015 forward a cache read address through memory request generator 1020 and through pipeline latency elements 1025 to cache data store 1030" such that "[c]ache data store 1030 responsively outputs on lines 649 the texture values cached in the read location" (Col. 6, lines 42-47 – emphasis added). Clearly, Rivard is disclosing the transfer of graphical information into DRAM and outputting cached texture values, which simply fails to support the Examiner's allegation that "instructions stored in DRAM that are requested are basically stored as data" (emphasis added). Therefore, the disclosure of transferring graphical information into DRAM and outputting cached texture values, as in Rivard, simply fails to suggest "sending an instruction request to video memory, where a texture module sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

Furthermore, appellant again asserts that the dictionary.com reference provided by the Examiner merely defines an instruction as "a command given to a computer to carry out a particular operation." In addition, the excerpts from Rivard disclose that the "memory request generator 1020 generates memory requests for all misses... and forwards the requests to DRAM 655 for

information retrieval" (Col. 6, lines 50-53 – emphasis added). However, generating and forwarding memory requests to DRAM for information retrieval, as in Rivard, fails to support the Examiner's allegation that "[m]emory requests/read requests for all misses that are forwarded to DRAM are instruction requests based on which DRAM sends back information" (emphasis added). Clearly, generating and forwarding memory requests to DRAM for information retrieval, as in Rivard, simply fails to specifically teach "sending an instruction request to video memory, where a texture module sends the instruction request to the video memory" (emphasis added), as claimed by appellant. Applicant emphasizes that the memory request of Rivard fails to meet appellant's claimed "instruction request," as claimed by appellant.

Additionally, with respect to independent Claims 25 and 26, the Examiner has relied on Figure 6 and Figure 10; Col. 4, lines 46-57; Col. 6, lines 45-67; and Col. 7, lines 1-10 from Rivard (reproduced above) to make a prior art showing of appellant's claimed "receiving instructions from the video memory in response to the instruction request" (see this or similar, but not necessarily identical language in the foregoing independent claims).

Specifically, the Examiner has argued that "DRAM returns memory data to cache data store and memory data resolver component of texel cache system, which further sends this information to the texture mapping stage; [where] the texture mapping stage and the texel cache system together are considered the cache module, so the information/memory data is passed between the components of the texture module."

Appellant respectfully disagrees, and points out that in the second paragraph on Page 7 of the Office Action mailed 03/20/2007, the Examiner has stated that "Rivard does not explicitly teach to combine texture mapping stage and texel cache system to form a texture module," and thus, Rivard simply fails to suggest "receiving instructions from the video memory in response to the instruction request" (emphasis added), as claimed by appellant.

In addition, the excerpts from Rivard relied upon by the Examiner merely teach that "DRAM 655 returns the **memory data** on bus 660 to cache data store and memory data resolver 1030" (emphasis added). However, merely returning memory data to a cache data store, as in Rivard,

simply fails to teach "receiving instructions from the video memory in response to the instruction request" (emphasis added), as claimed by appellant.

Furthermore, appellant respectfully asserts that Rivard teaches that "[t]exture mapping block 645 sends information via bus 647 to texel cache system 650, and texel cache system 650 sends information via bus 649 back to texture mapping block 645" (Col. 4, lines 54-57). Additionally, Rivard discloses that "[t]exel sample address computation block 1005 forwards the higher resolution sample points A-D to cache tag block 1010 and forwards the lower resolution sample points E-H to cache tag block 1015" and "[i]f a miss occurs, then cache tag blocks 1010 and 1015 forward a cache write address to memory request generator 1020" which "generates memory requests for all misses... and forwards the requests to DRAM 655 for information retrieval" (Col. 6, lines 33-36, and lines 48-53 – emphasis added). Further, Rivard discloses that "DRAM 655 returns the memory data on bus 660 to cache data store and memory data resolver 1030, which stores the memory data at the write address" (Col. 6, lines 53-56 – emphasis added).

However, the disclosure of a memory request generator that generates memory requests for all misses and forwards the requests to DRAM for information retrieval, and that the DRAM then returns the memory data which is stored at the write address, as in Rivard, simply fails to even suggest "receiving instructions from the video memory in response to the instruction request" (emphasis added), as claimed by appellant. Clearly, returning memory data which is stored at a write address after a cache miss, as in Rivard, fails to meet "receiving instructions from the video memory in response to the instruction request" (emphasis added), as claimed by appellant. Appellant emphasizes that the memory data from the DRAM, as disclosed in Rivard, fails to teach or suggest "instructions," in the manner as claimed by appellant.

Furthermore, in the Office Action mailed 09/20/2007, on Page 7, lines 3-7; Page 5, lines 9-17; Page 6, line 10 to Page 7, line 4; and Page 8, lines 4-8, the Examiner has argued the following:

> "In response to [appellant's] arguments against the references individually, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986)." (Page 7, lines 3-7)

"Rivard further teaches receiving instructions from the video memory in response to the instruction request utilizing the texture module in the graphics pipeline (it is interpreted that instructions stored in DRAM that are requested are basically stored as data, and Rivard teaches to send back data in some form in response to the memory request generated as a result of a miss; DRAM returns memory data to cache data store and memory data resolver component of texel cache system, which further sends this information to the texture mapping stage; the texture mapping stage and the texel cache system together are considered as texture module, so the information/memory data is passed between the components of the texture module)." (Page 5, lines 9-17)

"The examiner further interprets that Rivard also teaches the arrival of memory data and associated instructions at the cache data store and memory data resolver component of the text cache system is coordinated by pipeline latency elements (col. 7 lines 4-7; memory data has associated instructions are returned by DRAM, though the data has it's associated instructions). Therefore, the examiner brings in the Wang reference that suggests a graphics subunit (texture module) in a graphics hardware system that supplies data and control signals to local frame buffer memory for executing a series of display instructions (Fig. 1, col. 5 lines 38-67; the computer memory includes the local frame buffer memory, which corresponds to the DRAM; based on the control signals send by the graphics hardware system, the frame buffer returns the polygon display instructions that includes the texture data, so that the graphics subunit executes the display instructions using this data). Therefore, it would have been obvious to one of ordinary skill in art at the time of present invention to have the memory return instructions as taught by Wang to the texture module of Rivard because these instructions are needed to render the graphics primitives to be displayed on the display device (col. 5 lines 43-45 and lines 63-67)." (Page 6, line 10 to Page 7, line 4)

"However, the examiner interprets that Rivard teaches to send a request to DRAM via a texture module, and based on this request, DRAM sends back data and it's associated information to the texture module (see the arguments above for details). Nonetheless, Rivard does not explicitly state that the data and it's associated instructions are returned by DRAM, though the data has it's associated functions." (Page 8, lines 4-8)

First, appellant notes that the Examiner's arguments on Page 7, lines 5-21; and Page 8, lines 9-20 in the Office Action mailed 09/20/2007 are substantially similar to the arguments made by the Examiner on Page 6, line 10 to Page 7, line 4 excerpted above. Further, on Page 7, line 21 the Examiner has additionally citied "*In re Lockhart*, 90 USPQ (CCPA 1951)." Second, appellant has clearly responded to such arguments above regarding how the excerpts repeatedly relied upon by the Examiner fail to meet or suggest appellant's claimed "receiving instructions from the video memory in response to the instruction request" (emphasis added), as claimed by appellant.

Further, appellant respectfully asserts that the excerpts from Rivard relied upon by the Examiner merely teach that "[m]emory request generator 1020 generates memory requests for all misses... and forwards the requests to DRAM 655 for information retrieval" such that "DRAM 655 returns the memory data on bus 660 to cache data store and memory data resolver 1030" (Col. 6, lines 50-55 – emphasis added). However, Rivard's disclosure that memory requests are forwarded to DRAM for information retrieval, where the DRAM returns the memory data, fails to support the Examiner's allegation that "Rivard further teaches receiving instructions from the video memory in response to the instruction request utilizing the texture module in the graphics pipeline" (emphasis added). Clearly, forwarding memory requests to DRAM for information retrieval, as in Rivard, simply fails to specifically teach "receiving instructions from the video memory in response to the instruction request" (emphasis added), as claimed by appellant.

In addition, appellant respectfully asserts that the excerpts from Rivard relied upon by the Examiner teach that "[b]ecause memory request generator 1020 is between cache tag blocks 1010, 1015 and cache data store 1030, generator 1020 can perform DRAM 655 memory requests before the address and instruction information reach cache data store and memory data resolver 1030" (Col. 6, lines 62-66 – emphasis added). Furthermore, Wang teaches "a 3D graphics subunit 109 for executing a series of display instructions found within a display list stored in computer memory" and that "[m]any of the polygon display instructions include texture data to be displayed within the polygon" (Col. 5, lines 40-48 – emphasis added). However, the mere disclosure of memory request generator 1020 performing DRAM memory requests before the instruction information reaches the cache data store, as in Rivard, in addition to the disclosure of executing display instructions that may include texture data, as in Wang, simply fails to suggest "receiving instructions from the video memory in response to the instruction request" (emphasis added), as claimed by appellant. Clearly, performing memory requests before instruction information reaches the cache data store, as in Rivard, in addition to the disclosure of executing display instructions that may include texture data, as in Wang, simply fails to even suggest an "instruction request," much less "receiving instructions from the video memory in response to the instruction request" (emphasis added), as claimed by appellant.

Further, appellant respectfully asserts that Rivard's disclosure of a "graphics accelerator system 635 [that] includes pipeline latency elements 1025 to coordinate arrival of the memory data and

of the associated instructions at cache data store and memory data resolver 1030" (Col. 7, lines 4-7) clearly *teaches away* from the Examiner's allegation that it would be obvious to "have the memory return instructions as taught by Wang to the texture module of Rivard because these instructions are needed to render the graphics primitives to be displayed on the display device," as noted by the Examiner, since the "pipeline latency elements... coordinate arrival of the memory data and... the associated instructions" (emphasis added). Clearly, as argued above, it would not be obvious from the teachings of Rivard and Wang to "receiv[e] instructions from the video memory in response to the instruction request" (emphasis added), as claimed by appellant.

Still yet, appellant agrees with the Examiner's statement that "Rivard does not explicitly state that the data and it's associated instructions are returned by DRAM." For example, appellant asserts that the figures and excerpts from Rivard relied upon by the Examiner teach that texel cache system 650 "includes cache tags 1010, cache tags 1015, a memory request generator 1020, pipeline latency elements 1025 and cache data store and memory data resolver 1030" (see Figure 10; Col. 6, lines 22-26 – emphasis added) where the "[m]emory request generator 1020 generates memory requests for all misses... and forwards the requests to DRAM 655 for information retrieval" (Col. 6, lines 50-53 – emphasis added). Clearly, a texel cache system 650 including a memory request generator 1020 that generates memory requests that are forwarded to DRAM 655 for information retrieval, as in Rivard, simply fails to suggest "receiving instructions from the video memory in response to the instruction request" (emphasis added), as claimed by appellant.

Again, appellant respectfully asserts that at least the first and third elements of the *prima facie* case of obviousness have not been met, since it would be *unobvious* to combine the references, as noted above, and the prior art excerpts, as relied upon by the Examiner, fail to teach or suggest all of the claim limitations, as noted above.

Group #3: Claim 28

Moreover, with respect to independent Claim 28, the Examiner has relied on Figures 6 and 10; Col. 4, lines 46-57; Col. 6, lines 45-67; and Col. 7, lines 1-10 in Rivard to make a prior art

showing of appellant's claimed "sending an instruction request to video memory, where the texture module sends the instruction request to the video memory."

Appellant respectfully asserts that figures from Rivard relied upon by the Examiner only shows a block diagram of a computer system, and that the excerpts relied on by the Examiner simply disclose that "the cache tag blocks 1010 and 1015 forward a cache write address to memory request generator 1020" where "[m]emory request generator 1020 generates memory requests for all misses." Clearly, generating a memory request for all misses, as in Rivard, does not meet appellant's claimed "sending an instruction request" (emphasis added), as claimed.

In the Office Action mailed 03/20/2007, the Examiner has argued that "Rivard teaches this limitation" and to "refer to the rejection of claim 1... regarding sending an instruction request to video memory, where a texture module in a graphics pipeline sends the instruction request to the video memory." Appellant disagrees with the Examiner's arguments and asserts that, for substantially the same reasons as those argued above with respect to at least some of the independent claims, Rivard fails to even suggest "sending an instruction request," as appellant claims. For example, appellant respectfully points out that Rivard merely teaches that "the memory request generator is coupled between the cache tag block and the cache data store for performing a memory request before an address and instruction information associated with the needed texels reach the cache data store" (Col. 10, lines 48-52 - emphasis added). Clearly, performing a memory request before the address and instruction information reach the cache data store, as in Rivard (see excerpts above), fails to teach "sending an instruction request" (emphasis added), as claimed by appellant.

Furthermore, with respect to Claim 28 and the Examiner's reliance upon the rejection of Claim 1, the Examiner has argued that "the definition of instruction provided by dictionary.com, which defines instructions as a command given to a computer to carry out a particular operation and can contain data to be used in the operation; here DRAM sends memory data based on the memory requests/read requests from the texture module." Further, the Examiner has argued that "[m]emory requests/read requests act as an instruction to DRAM, which performs a particular function based on the request."

Appellant respectfully disagrees with the Examiner's arguments and asserts that the dictionary.com reference provided by the Examiner merely defines an instruction as "a command given to a computer to carry out a particular operation." Appellant respectfully asserts that Rivard merely discloses that the "cache tag blocks 1010 and 1015 forward a cache write address to memory request generator 1020" which "generates memory requests for all misses... and forwards the requests to DRAM 655 for information retrieval" (Col. 6, lines 33-36, and lines 48-53 – emphasis added). However, the disclosure of the memory request generator generating a memory request and forwarding the request to DRAM for information retrieval, as in Rivard, simply fails to suggest "sending an instruction request to video memory, where the texture module sends the instruction request to the video memory" (emphasis added), as claimed by appellant. Clearly, generating and forwarding a memory request to DRAM, as in Rivard, fails to suggest "sending an instruction request" (emphasis added), in the manner as claimed by appellant.

In the Office Action mailed 09/20/2007, on Page 2, line 19 to Page 3, line 19; and Page 4, lines 3-7, the Examiner has argued the following:

> "However, the examiner interprets that Rivard (Fig. 6, Fig. 10, col. 4 lines 46-57, col. 6 lines 45-67, col. 7 lines 1-10) teaches sending an instruction request to video memory, where a texture module in a graphics pipeline sends the instruction request to the video memory (a texture mapping stage sends information to texel cache system; the texture mapping stage and the texel cache system together are considered as texture module, so the information is passed between the components of the texture module; texel cache system comprising the memory request generator, pipeline latency elements, and cache data store and memory data resolver forwards the memory requests/read requests to DRAM for information retrieval. [T]he examiner interprets that Rivard discloses sending an instruction request to video memory by generating memory request for all misses, and forwards the requests to DRAM for information retrieval (Fig. 6, Fig. 10, col. 4 lines 46-57, col. 6 lines 45-67, col. 7 lines 1-10). It is interpreted that an instruction request could be considered as merely a request because it is basically a request that requires DRAM to perform necessary operation. The examiner also interprets that instructions stored in DRAM that are requested are basically stored as data. It is further interpreted that memory requests/read requests act as an instruction to DRAM based on which DRAM performs a particular function. Memory requests/read requests for all misses that are forwarded to DRAM are instruction requests based on which DRAM sends back information; the examiner relies on the definition of instruction provided by dictionary.com, which defines instructions as a command given to a computer to carry out a particular operation and can contain data to be used in the operation; here DRAM sends memory based on the memory requests/read requests from the texture module." (Page 2, line 19 to Page 3, line 19)

> "In response to [appellant's] arguments against the references individually, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986)." (Page 4, lines 3-7)

First, appellant notes that the Examiner's arguments on Page 4, line 8 to Page 5, line 8 in the Office Action mailed 09/20/2007 are substantially similar to the arguments made by the Examiner on Page 2, line 19 to Page 3, line 19 excerpted above. Second, appellant has clearly responded to such arguments above regarding how the excerpts repeatedly relied upon by the Examiner fail to meet or suggest appellant's claimed "sending an instruction request to video memory, where the texture module sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

Further, appellant respectfully disagrees with the Examiner's arguments and asserts that the figures and excerpts from Rivard relied upon by the Examiner merely disclose that "the texture mode indicates a 16-bit per pixel texture lookup" in addition to "satisfy[ing] two lookup requests with a single read request" (Col. 6, lines 56-61 – emphasis added). Additionally, the excerpts from Rivard teach that "pipeline latency elements 1025… coordinate arrival of the memory data and of the associated instructions" (Col. 7, lines 3-7 – emphasis added) such that "the memory request generator… perform[s] a memory request before an address and instruction information associated with the needed texels reach the cache data store" (Col. 10, lines 48-52 – emphasis added). Clearly, the texture mode indicating a texture lookup, in addition to the memory request generator performing a memory request, as in Rivard, simply fails to support the Examiner's allegation that "a texture module in a graphics pipeline sends the instruction request to the video memory" (emphasis added), and especially does not teach "sending an instruction request to video memory, where the texture module sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

In addition, Rivard's disclosure that the "cache tag blocks 1010 and 1015 forward a cache write address to memory request generator 1020," which "generates memory requests for all misses… and forwards the requests to DRAM 655 for information retrieval" (Col. 6, lines 48-53 – emphasis added). However, the disclosure of the memory request generator generating a

memory request and forwarding the request to DRAM for information retrieval, as in Rivard, fails to support the Examiner's allegation "that Rivard discloses sending an instruction request to video memory by generating memory request for all misses, and forwards the requests to DRAM for information retrieval" (emphasis added). Clearly, generating a memory request and forwarding the request to DRAM for information retrieval, as in Rivard, simply fails to specifically teach "sending an instruction request to video memory, where the texture module sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

Still yet, appellant respectfully asserts that the excerpts from Rivard relied upon by the Examiner disclose that the "[g]raphics application program 670 transfers graphical information from data storage 625 into DRAM 655 for local storage" (Col. 4, lines 46-48 – emphasis added). Additionally, the excerpts disclose that "[i]f a hit occurs, then cache tag blocks 1010 and 1015 forward a cache read address through memory request generator 1020 and through pipeline latency elements 1025 to cache data store 1030" such that "[c]ache data store 1030 responsively outputs on lines 649 the texture values cached in the read location" (Col. 6, lines 42-47 – emphasis added). Clearly, Rivard is disclosing the transfer of graphical information into DRAM and outputting cached texture values, which simply fails to support the Examiner's allegation that "instructions stored in DRAM that are requested are basically stored as data" (emphasis added). Therefore, the disclosure of transferring graphical information into DRAM and outputting cached texture values, as in Rivard, simply fails to suggest "sending an instruction request to video memory, where the texture module sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

Furthermore, appellant again asserts that the dictionary.com reference provided by the Examiner merely defines an instruction as "a command given to a computer to carry out a particular operation." In addition, the excerpts from Rivard disclose that the "memory request generator 1020 generates memory requests for all misses... and forwards the requests to DRAM 655 for information retrieval" (Col. 6, lines 50-53 – emphasis added). However, generating and forwarding memory requests to DRAM for information retrieval, as in Rivard, fails to support the Examiner's allegation that "[m]emory requests/read requests for all misses that are forwarded to DRAM are instruction requests based on which DRAM sends back information" (emphasis added). Clearly, generating and forwarding memory requests to DRAM for information

retrieval, as in Rivard, simply fails to specifically teach "sending an instruction request to video memory, where the texture module sends the instruction request to the video memory" (emphasis added), as claimed by appellant. Applicant emphasizes that the memory request of Rivard fails to meet appellant's claimed "instruction request," as claimed by appellant.

In addition, with respect to Claim 28, the Examiner has relied on Figures 6 and 10; Col. 4, lines 46-57; Col. 6, lines 45-67; and Col. 7, lines 1-10 from Rivard to make a prior art showing of appellant's claimed "receiving additional instructions from the video memory in response to the instruction request utilizing the texture module."

Specifically, the Examiner has argued that "Rivard... teaches that DRAM returns memory data to cache data store and memory data resolver component of texel cache system, which further sends this information to the texture mapping stage." Further, the Examiner has argued that "the texture mapping stage and the texel cache system together are considered as texture module, so the information is passed between the components of the texture module." In addition, the Examiner has argued that "[t]he texture mapping stage as shown in Fig. 6 also receives data (instructions) from the rasterizer module of the pipeline, and thus the instructions received from the DRAM via the texel cache system are considered to be the additional instructions."

Appellant respectfully disagrees with the Examiner's arguments and asserts that Rivard merely discloses that "DRAM 655 returns the memory data on bus 660 to cache data store and memory data resolver 1030, which stores the memory data at the write address" (Col. 6, lines 53-56 – emphasis added). However, merely storing memory data at the write address, as in Rivard, simply fails to suggest "receiving additional instructions from the video memory in response to the instruction request utilizing the texture module" (emphasis added), as claimed by appellant. Clearly, storing memory data, as in Rivard, fails to suggest "receiving additional instructions," in the manner as claimed by appellant.

Second, in response to the Examiner's argument that "the texture mapping stage and the texel cache system together are considered as texture module," appellant respectfully disagrees and asserts that Rivard merely teaches that "[g]raphics accelerator system 635 includes graphics pipeline stages 640 including a texture mapping stage 645 for mapping texture information to the

graphics information received from graphics application program 670 and for maintaining the texel information in a texel cache system 650" where the "[t]exture mapping block 645 sends information via bus 647 to texel cache system 650, and texel cache system 650 sends information via bus 649 back to texture mapping block 645" (Col. 4, lines 49-57 – emphasis added). Clearly, the texture mapping stage is separate from the texture cache system, as in Rivard, which fails to suggest "receiving additional instructions from the video memory in response to the instruction request utilizing the texture module" (emphasis added), as claimed by appellant.

Third, in response to the Examiner's argument that "Fig. 6 also receives data (instructions) from the rasterizer module of the pipeline, and thus the instructions received from the DRAM via the texel cache system are considered to be the additional instructions," appellant respectfully disagrees and asserts that Rivard merely teaches that "the memory request generator is coupled between the cache tag block and the cache data store for performing a memory request before an address and instruction information associated with the needed texels reach the cache data store" (emphasis added). Clearly, coordinating the arrival of "memory data", where "the data is conveniently aligned in the texture map" (emphasis added), as in Rivard (see excerpts above), fails to teach "receiving additional instructions" (emphasis added), as claimed by appellant. In addition, appellant respectfully asserts that simply nowhere in the description of Figure 6 is there any disclosure that "Fig. 6 also receives data (instructions) from the rasterizer module of the pipeline," as noted by the Examiner. Thus, Rivard simply fails to meet appellant's claimed "receiving additional instructions... utilizing the texture module," as claimed.

Furthermore, in the Office Action mailed 09/20/2007, on Page 9, line 4 to Page 10, line 5, the Examiner has argued the following:

> "In response to [appellant's] arguments against references individually, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986).
>
> In this case, the examiner interprets that Rivard teaches a texture mapping stage as shown in Fig. 6 receiving data (instructions) from the rasterizer module of the pipeline and then receiving data (data and it's associated instructions) from the DRAM. The instructions received from the rasterizer module are considered to be the first instructions, and thus the instructions received from the DRAM via the texel cache system are considered to be the additional instructions.

> Since Rivard does not explicitly state that the data and it's associated instructions are returned by DRAM, though the data has it's associated instructions, the examiner relies on the Wang reference that suggests a graphics subunit (texture module) in a graphics hardware system that supplies data and control signals to local frame buffer memory for executing a series of display instructions (Fig. 1, col. 5 lines 38-67; the computer memory includes the local frame buffer memory, which corresponds to the DRAM; based on the control signals send by the graphics hardware system, the display list stored in the frame buffer returns the polygon display instructions that includes the texture data, so that the graphics subunit executes the display instructions using this data). Therefore, it would have been obvious to one of ordinary skill in art at the time of present invention to have the memory return instructions as taught by Wang to the texture module of Rivard because these instructions are needed to render the graphics primitives to be displayed on the display device (col. 5 lines 43-45 and lines 63-67)." (Page 9, line 4 to Page 10, line 5)

First, appellant has clearly argued above how the excerpts repeatedly relied upon by the Examiner fail to meet or suggest appellant's claimed "receiving additional instructions from the video memory in response to the instruction request utilizing the texture module" (emphasis added), as claimed by appellant.

Further, appellant respectfully asserts that Figure 6 from the Rivard reference merely teaches that "[g]raphics application program 670 transfers graphical information from data storage 625 into DRAM 655 for local storage" and that "[g]raphics accelerator system 635 includes graphics pipeline stages 640 including a texture mapping stage 645 for mapping texture information to the graphics information received from graphics application program 670 and for maintaining the texel information in a texel cache system 650" (Col. 4, lines 46-54 – emphasis added). Additionally, Figure 6 teaches that "[t]exture mapping block 645 sends information via bus 647 to texel cache system 650, and texel cache system 650 sends information via bus 649 back to texture mapping block 645" (Col. 4, lines 54-57 – emphasis added). However, Rivard's teaching that graphic pipeline stages 640 includes a texture mapping stage 645 for mapping texture information and maintaining the texel information in a texel cache system 650, such that the texel cache system 650 sends information to texture mapping block 645, simply fails to support the Examiner's allegation that "Rivard teaches a texture mapping stage as shown in Fig. 6 receiving data (instructions) from the rasterizer module of the pipeline and then receiving data (data and it's associated instructions) from the DRAM" (emphasis added). Clearly, a texture mapping stage 645 for mapping texture information and maintaining the texel information in a texel cache system 650, such that the texel cache system 650 sends information to texture

mapping block 645, as in Rivard, simply fails to even suggest "receiving additional instructions from the video memory in response to the instruction request utilizing the texture module" (emphasis added), as claimed by appellant.

Further, appellant respectfully asserts that the excerpts from Rivard relied upon by the Examiner merely teach that "[m]emory request generator 1020 generates memory requests for all misses... and forwards the requests to DRAM 655 for information retrieval" such that "DRAM 655 returns the memory data on bus 660 to cache data store and memory data resolver 1030" (Col. 6, lines 50-55 – emphasis added). Clearly, forwarding memory requests to DRAM for information retrieval, as in Rivard, simply fails to specifically teach "receiving additional instructions from the video memory in response to the instruction request utilizing the texture module" (emphasis added), as claimed by appellant.

In addition, appellant respectfully asserts that the excerpts from Rivard relied upon by the Examiner teach that "[b]ecause memory request generator 1020 is between cache tag blocks 1010, 1015 and cache data store 1030, generator 1020 can perform DRAM 655 memory requests before the address and instruction information reach cache data store and memory data resolver 1030" (Col. 6, lines 62-66 – emphasis added). Furthermore, Wang teaches "a 3D graphics subunit 109 for executing a series of display instructions found within a display list stored in computer memory" and that "[m]any of the polygon display instructions include texture data to be displayed within the polygon" (Col. 5, lines 40-48 – emphasis added). However, the mere disclosure of memory request generator 1020 performing DRAM memory requests before the instruction information reaches the cache data store, as in Rivard, in addition to the disclosure of executing display instructions that may include texture data, as in Wang, simply fails to suggest "receiving additional instructions from the video memory in response to the instruction request utilizing the texture module" (emphasis added), as claimed by appellant. Clearly, performing memory requests before instruction information reaches the cache data store, as in Rivard, in addition to the disclosure of executing display instructions that may include texture data, as in Wang, simply fails to even suggest an "instruction request," much less "receiving additional instructions from the video memory in response to the instruction request utilizing the texture module" (emphasis added), as claimed by appellant.

Further, appellant respectfully asserts that Rivard's disclosure of a "graphics accelerator system 635 [that] includes pipeline latency elements 1025 to coordinate arrival of the memory data and of the associated instructions at cache data store and memory data resolver 1030" (Col. 7, lines 4-7) clearly *teaches away* from the Examiner's allegation that it would be obvious to "have the memory return instructions as taught by Wang to the texture module of Rivard because these instructions are needed to render the graphics primitives to be displayed on the display device," as noted by the Examiner, since in Rivard the "pipeline latency elements... coordinate arrival of the memory data and... the associated instructions" (emphasis added). Clearly, as argued above, it would not be obvious from the teachings of Rivard and Wang to "receiving additional instructions from the video memory in response to the instruction request utilizing the texture module" (emphasis added), as claimed by appellant.

Still yet, appellant agrees with the Examiner's statement that "Rivard does not explicitly state that the data and it's associated instructions are returned by DRAM." For example, appellant asserts that the figures and excerpts from Rivard relied upon by the Examiner teach that texel cache system 650 "includes cache tags 1010, cache tags 1015, a memory request generator 1020, pipeline latency elements 1025 and cache data store and memory data resolver 1030" (see Figure 10; Col. 6, lines 22-26 – emphasis added) where the "[m]emory request generator 1020 generates memory requests for all misses... and forwards the requests to DRAM 655 for information retrieval" (Col. 6, lines 50-53 – emphasis added). Clearly, a texel cache system 650 including a memory request generator 1020 that generates memory requests that are forwarded to DRAM 655 for information retrieval, as in Rivard, simply fails to suggest "receiving additional instructions from the video memory in response to the instruction request utilizing the texture module" (emphasis added), as claimed by appellant.

Again, appellant respectfully asserts that at least the first and third elements of the *prima facie* case of obviousness have not been met, since it would be *unobvious* to combine the references, as noted above, and the prior art excerpts, as relied upon by the Examiner, fail to teach or suggest all of the claim limitations, as noted above.

Group #4: Claim 30

With respect to independent Claim 30, the Examiner has relied on Figure 6 and Figure 10, in addition to Col. 4, lines 46-57; and Col. 6, line 45-Col. 7, line 10 (excerpted below) from Rivard to make a prior art showing of appellant's claimed "sending an instruction request to video memory in a graphics pipeline, where a cache in the graphics pipeline sends the instruction request to the video memory."

> "Graphics application program 670 transfers graphical information from data storage 625 into DRAM 655 for local storage. Graphics accelerator system 635 includes graphics pipeline stages 640 including a texture mapping stage 645 for mapping texture information to the graphics information received from graphics application program 670 and for maintaining the texel information in a texel cache system 650. Texture mapping block 645 sends information via bus 647 to texel cache system 650, and texel cache system 650 sends information via bus 649 back to texture mapping block 645." (Col. 4, lines 46-57 – emphasis added)
>
> "Cache data store 1030 responsively outputs on lines 649 the texture values cached in the read location.
>
> If a miss occurs, then cache tag blocks 1010 and 1015 forward a cache write address to memory request generator 1020. Memory request generator 1020 generates memory requests for all misses (up to four for bi-linear sampling and up to eight for tri-linear sampling), and forwards the requests to DRAM 655 for information retrieval. DRAM 655 returns the memory data on bus 660 to cache data store and memory data resolver 1030, which stores the memory data at the write address. If the interface to DRAM 655 is 32-bits wide, the texture mode indicates a 16-bit per pixel texture lookup and the data is conveniently aligned in the texture map, then it is possible to satisfy two lookup requests with a single read request. However, if the data is not aligned, then two read requests are needed.
>
> Because memory request generator 1020 is between cache tag blocks 1010, 1015 and cache data store 1030, generator 1020 can perform DRAM 655 memory requests before the address and instruction information reach cache data store and memory data resolver 1030. Once memory requests are generated, there is, depending on the DRAM design, a constant latency of about five to ten clock cycles (or possibly more) which includes time for exiting and reentering the graphics pipeline hardware, to effect a page hit. Therefore, graphics accelerator system 635 includes pipeline latency elements 1025 to coordinate arrival of the memory data and of the associated instructions at cache data store and memory data resolver 1030." (Col. 6, line 45-Col. 7, line 10 – emphasis added)

Appellant respectfully asserts that the figures relied on by the Examiner only generally illustrate a computer system and a block diagram detailing the graphics accelerator system showing the pipeline latency elements 1025. In addition, the excerpts relied on by the Examiner merely teach that the "[m]emory request generator 1020 generates memory requests for all misses (up to four

for bi-linear sampling and up to eight for tri-linear sampling), and forwards the requests to DRAM 655 for information retrieval" (emphasis added). However, disclosing that a memory request is generated for all misses, as in Rivard, fails to teach "sending an instruction request to video memory in a graphics pipeline, where a cache in the graphics pipeline sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

Appellant notes that the Examiner has argued that "the texture mapping stage and the texel cache system together are considered the texture module." Appellant respectfully disagrees and asserts that Rivard merely suggests that the "[g]raphics accelerator system 635 includes graphics pipeline stages 640 including a texture mapping stage 645 for mapping texture information to the graphics information received from graphics application program 670 and for maintaining the texel information in a texel cache system 650" where the "[t]exture mapping block 645 sends information via bus 647 to texel cache system 650, and texel cache system 650 sends information via bus 649 back to texture mapping block 645" (Col. 4, lines 49-57 – emphasis added). Clearly, the texture mapping stage **is separate from** the texture cache system, as clearly disclosed in Rivard, and thus fails to suggest "sending an instruction request to video memory in a graphics pipeline, where a cache in the graphics pipeline sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

In addition, the Examiner has argued that "memory requests/read requests for all misses that are forwarded to DRAM are instruction requests based on which DRAM sends back information… [where] the DRAM sends memory data based on the memory requests/read requests from the texture module." Further, the Examiner has argued that "[m]emory requests/read requests act as an instruction to DRAM, which performs a particular function based on the request."

Appellant respectfully disagrees, and asserts that Rivard merely discloses that the "[t]exture mapping block 645 sends information via bus 647 to texel cache system 650, and texel cache system 650 sends information via bus 649 back to texture mapping block 645" (Col. 4, lines 54-57). Additionally, Rivard discloses that "[t]exel sample address computation block 1005 forwards the higher resolution sample points A-D to cache tag block 1010 and forwards the lower resolution sample points E-H to cache tag block 1015" and "[i]f a miss occurs, then cache tag blocks 1010 and 1015 forward a cache write address to memory request generator 1020"

which "generates memory requests for all misses... and forwards the requests to DRAM 655 for information retrieval" (Col. 6, lines 33-36, and lines 48-53 – emphasis added).

However, Rivard's disclosure that the memory request generator 1020, which is included in the texel cache system (see Figure 10), generates memory requests for all misses and forwards the requests to DRAM for information retrieval, simply fails to even suggest "sending an instruction request to video memory in a graphics pipeline, where a cache in the graphics pipeline sends the instruction request to the video memory" (emphasis added), as claimed by appellant. Moreover, the texel sample address computation block (included in texture mapping block 645) which forwards sample points to the cache tag blocks 1010 and 1015 (included in texel cache system 650), as disclosed in Rivard, fails to meet "sending an instruction request to video memory in a graphics pipeline, where a cache in the graphics pipeline sends the instruction request to the video memory" (emphasis added), as claimed by appellant. Furthermore, the memory request to the DRAM, as in Rivard, fails to suggest "an instruction request," in the manner as claimed by appellant.

In addition, appellant respectfully asserts that the following excerpts from Rivard further demonstrate that the Rivard fails to disclose appellant's claimed "sending an instruction request to video memory in a graphics pipeline, where a cache in the graphics pipeline sends the instruction request to the video memory."

> "If the interface to DRAM 655 is 32-bits wide, the texture mode indicates a 16-bit per pixel texture lookup and the data is conveniently aligned in the texture map, then it is possible to satisfy two lookup requests with a single read request. However, if the data is not aligned, then two read requests are needed." (Col. 6, lines 56-61 - emphasis added)
>
> "Therefore, graphics accelerator system 635 includes pipeline latency elements 1025 to coordinate arrival of the memory data and of the associated instructions at cache data store and memory data resolver 1030." (Col. 7, lines 3-7 - emphasis added)
>
> "...the memory request generator is coupled between the cache tag block and the cache data store for performing a memory request before an address and instruction information associated with the needed texels reach the cache data store." (Col. 10, lines 48-52 - emphasis added)

First, appellant respectfully points out that Rivard merely teaches that "the memory request generator is coupled between the cache tag block and the cache data store for performing a memory request before an address and instruction information associated with the needed texels reach the cache data store" (emphasis added). Clearly, coordinating the arrival of "memory data", where "the data is conveniently aligned in the texture map" (emphasis added), as in Rivard (see excerpts above), fails to teach "sending an instruction request" (emphasis added), as claimed by appellant.

Second, appellant respectfully asserts that Rivard simply discloses that a memory request is performed before an address and instruction information associated with the needed texels reach the cache data store (see excerpts above). Thus, the memory data in Rivard simply relates to texel data, which clearly fails to suggest "sending an instruction request to video memory in a graphics pipeline, where a cache in the graphics pipeline sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

Furthermore, the Examiner has argued that "the definition of instruction provided by dictionary.com, which defines instructions as a command given to a computer to carry out a particular operation and can contain data to be used in the operation; here DRAM sends memory data based on the memory requests/read requests from the texture module." Further, the Examiner has argued that "[m]emory requests/read requests act as an instruction to DRAM, which performs a particular function based on the request."

Appellant respectfully disagrees with the Examiner's arguments and asserts that the dictionary.com reference provided by the Examiner merely defines an instruction as "a command given to a computer to carry out a particular operation." Appellant respectfully asserts that Rivard merely discloses that the "cache tag blocks 1010 and 1015 forward a cache write address to memory request generator 1020" which "generates memory requests for all misses... and forwards the requests to DRAM 655 for information retrieval" (Col. 6, lines 33-36, and lines 48-53 – emphasis added). However, the disclosure of the memory request generator generating a memory request and forwarding the request to DRAM for information retrieval, as in Rivard, simply fails to suggest "sending an instruction request to video memory in a graphics pipeline, where a cache in the graphics pipeline sends the instruction request to the video memory"

(emphasis added), as claimed by appellant. Clearly, generating and forwarding a memory request to DRAM, as in Rivard, fails to suggest "sending an instruction request," particularly where "instructions [are received]…in response to the instruction request" (emphasis added), in the context as claimed by appellant.

In the Office Action mailed 09/20/2007, on Page 2, line 19 to Page 3, line 19; and Page 4, lines 3-7, the Examiner has argued the following:

> "However, the examiner interprets that Rivard (Fig. 6, Fig. 10, col. 4 lines 46-57, col. 6 lines 45-67, col. 7 lines 1-10) teaches sending an instruction request to video memory, where a texture module in a graphics pipeline sends the instruction request to the video memory (a texture mapping stage sends information to texel cache system; the texture mapping stage and the texel cache system together are considered as texture module, so the information is passed between the components of the texture module; texel cache system comprising the memory request generator, pipeline latency elements, and cache data store and memory data resolver forwards the memory requests/read requests to DRAM for information retrieval. [T]he examiner interprets that Rivard discloses sending an instruction request to video memory by generating memory request for all misses, and forwards the requests to DRAM for information retrieval (Fig. 6, Fig. 10, col. 4 lines 46-57, col. 6 lines 45-67, col. 7 lines 1-10). It is interpreted that an instruction request could be considered as merely a request because it is basically a request that requires DRAM to perform necessary operation. The examiner also interprets that instructions stored in DRAM that are requested are basically stored as data. It is further interpreted that memory requests/read requests act as an instruction to DRAM based on which DRAM performs a particular function. Memory requests/read requests for all misses that are forwarded to DRAM are instruction requests based on which DRAM sends back information; the examiner relies on the definition of instruction provided by dictionary.com, which defines instructions as a command given to a computer to carry out a particular operation and can contain data to be used in the operation; here DRAM sends memory based on the memory requests/read requests from the texture module." (Page 2, line 19 to Page 3, line 19)

> "In response to [appellant's] arguments against the references individually, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986)." (Page 4, lines 3-7)

First, appellant notes that the Examiner's arguments on Page 4, line 8 to Page 5, line 8 in the Office Action mailed 09/20/2007 are substantially similar to the arguments made by the Examiner on Page 2, line 19 to Page 3, line 19 excerpted above. Second, appellant has clearly responded to such arguments above regarding how the excerpts repeatedly relied upon by the

Examiner fail to meet or suggest appellant's claimed "sending an instruction request to video memory in a graphics pipeline, where a cache in the graphics pipeline sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

Further, appellant respectfully disagrees with the Examiner's arguments and asserts that the figures and excerpts from Rivard relied upon by the Examiner merely disclose that "the texture mode indicates a 16-bit per pixel texture lookup" in addition to "satisfy[ing] two lookup requests with a single read request" (Col. 6, lines 56-61 – emphasis added). Additionally, the excerpts from Rivard teach that "pipeline latency elements 1025… coordinate arrival of the memory data and of the associated instructions" (Col. 7, lines 3-7 – emphasis added) such that "the memory request generator… perform[s] a memory request before an address and instruction information associated with the needed texels reach the cache data store" (Col. 10, lines 48-52 – emphasis added). Clearly, the texture mode indicating a texture lookup, in addition to the memory request generator performing a memory request, as in Rivard, simply fails to support the Examiner's allegation that "a texture module in a graphics pipeline sends the instruction request to the video memory" (emphasis added), and especially does not teach "sending an instruction request to video memory in a graphics pipeline, where a cache in the graphics pipeline sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

In addition, Rivard's disclosure that the "cache tag blocks 1010 and 1015 forward a cache write address to memory request generator 1020," which "generates memory requests for all misses… and forwards the requests to DRAM 655 for information retrieval" (Col. 6, lines 48-53 – emphasis added). However, the disclosure of the memory request generator generating a memory request and forwarding the request to DRAM for information retrieval, as in Rivard, fails to support the Examiner's allegation "that Rivard discloses sending an instruction request to video memory by generating memory request for all misses, and forwards the requests to DRAM for information retrieval" (emphasis added). Clearly, generating a memory request and forwarding the request to DRAM for information retrieval, as in Rivard, simply fails to specifically teach "sending an instruction request to video memory in a graphics pipeline, where a cache in the graphics pipeline sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

Still yet, appellant respectfully asserts that the excerpts from Rivard relied upon by the Examiner disclose that the "[g]raphics application program 670 transfers graphical information from data storage 625 into DRAM 655 for local storage" (Col. 4, lines 46-48 – emphasis added). Additionally, the excerpts disclose that "[i]f a hit occurs, then cache tag blocks 1010 and 1015 forward a cache read address through memory request generator 1020 and through pipeline latency elements 1025 to cache data store 1030" such that "[c]ache data store 1030 responsively outputs on lines 649 the texture values cached in the read location" (Col. 6, lines 42-47 – emphasis added). Clearly, Rivard is disclosing the transfer of graphical information into DRAM and outputting cached texture values, which simply fails to support the Examiner's allegation that "instructions stored in DRAM that are requested are basically stored as data" (emphasis added). Therefore, the disclosure of transferring graphical information into DRAM and outputting cached texture values, as in Rivard, simply fails to suggest "sending an instruction request to video memory in a graphics pipeline, where a cache in the graphics pipeline sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

Furthermore, appellant again asserts that the dictionary.com reference provided by the Examiner merely defines an instruction as "a command given to a computer to carry out a particular operation." In addition, the excerpts from Rivard disclose that the "memory request generator 1020 generates memory requests for all misses... and forwards the requests to DRAM 655 for information retrieval" (Col. 6, lines 50-53 – emphasis added). However, generating and forwarding memory requests to DRAM for information retrieval, as in Rivard, fails to support the Examiner's allegation that "[m]emory requests/read requests for all misses that are forwarded to DRAM are instruction requests based on which DRAM sends back information" (emphasis added). Clearly, generating and forwarding memory requests to DRAM for information retrieval, as in Rivard, simply fails to specifically teach "sending an instruction request to video memory in a graphics pipeline, where a cache in the graphics pipeline sends the instruction request to the video memory" (emphasis added), as claimed by appellant. Applicant emphasizes that the memory request of Rivard fails to meet appellant's claimed "instruction request," as claimed by appellant.

Additionally, with respect to independent Claim 30, the Examiner has relied on Figure 6 and Figure 10; Col. 4, lines 46-57; Col. 6, lines 45-67; and Col. 7, lines 1-10 from Rivard

(reproduced above) to make a prior art showing of appellant's claimed "receiving instructions from the video memory in response to the instruction request for storage in the cache in the graphics pipeline."

The excerpts from Rivard relied upon by the Examiner merely teach that "DRAM 655 returns the **memory data** on bus 660 to cache data store and memory data resolver 1030" (emphasis added). However, merely returning memory data to a cache data store, as in Rivard, simply fails to teach "receiving instructions from the video memory in response to the instruction request for storage in the cache in the graphics pipeline" (emphasis added), as claimed by appellant.

Furthermore, appellant respectfully asserts that Rivard teaches that "[t]exture mapping block 645 sends information via bus 647 to texel cache system 650, and texel cache system 650 sends information via bus 649 back to texture mapping block 645" (Col. 4, lines 54-57) Additionally, Rivard discloses that "[t]exel sample address computation block 1005 forwards the higher resolution sample points A-D to cache tag block 1010 and forwards the lower resolution sample points E-H to cache tag block 1015" and "[i]f a miss occurs, then cache tag blocks 1010 and 1015 forward a cache write address to memory request generator 1020" which "generates memory requests for all misses... and forwards the requests to DRAM 655 for information retrieval" (Col. 6, lines 33-36, and lines 48-53 – emphasis added). Further, Rivard discloses that "DRAM 655 returns the memory data on bus 660 to cache data store and memory data resolver 1030, which stores the memory data at the write address" (Col. 6, lines 53-56 – emphasis added).

However, the disclosure of a memory request generator that generates memory requests for all misses and forwards the requests to DRAM for information retrieval, and that the DRAM then returns the memory data which is stored at the write address, as in Rivard, simply fails to even suggest "receiving instructions from the video memory in response to the instruction request for storage in the cache in the graphics pipeline" (emphasis added), as claimed by appellant. Clearly, returning memory data which is stored at a write address after a cache miss, as in Rivard, fails to meet "receiving instructions from the video memory in response to the instruction request for storage in the cache" (emphasis added), as claimed by appellant. Appellant emphasizes that the memory data from the DRAM, as disclosed in Rivard, fails to teach or suggest "instructions," in the manner as claimed by appellant.

Furthermore, in the Office Action mailed 09/20/2007, on Page 7, lines 3-7; Page 5, lines 9-17; Page 6, line 10 to Page 7, line 4; and Page 8, lines 4-8, the Examiner has argued the following:

> "In response to [appellant's] arguments against the references individually, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986)." (Page 7, lines 3-7)

> "Rivard further teaches receiving instructions from the video memory in response to the instruction request utilizing the texture module in the graphics pipeline (it is interpreted that instructions stored in DRAM that are requested are basically stored as data, and Rivard teaches to send back data in some form in response to the memory request generated as a result of a miss; DRAM returns memory data to cache data store and memory data resolver component of texel cache system, which further sends this information to the texture mapping stage; the texture mapping stage and the texel cache system together are considered as texture module, so the information/memory data is passed between the components of the texture module)." (Page 5, lines 9-17)

> "The examiner further interprets that Rivard also teaches the arrival of memory data and associated instructions at the cache data store and memory data resolver component of the text cache system is coordinated by pipeline latency elements (col. 7 lines 4-7; memory data has associated instructions are returned by DRAM, though the data has it's associated instructions). Therefore, the examiner brings in the Wang reference that suggests a graphics subunit (texture module) in a graphics hardware system that supplies data and control signals to local frame buffer memory for executing a series of display instructions (Fig. 1, col. 5 lines 38-67; the computer memory includes the local frame buffer memory, which corresponds to the DRAM; based on the control signals send by the graphics hardware system, the frame buffer returns the polygon display instructions that includes the texture data, so that the graphics subunit executes the display instructions using this data). Therefore, it would have been obvious to one of ordinary skill in art at the time of present invention to have the memory return instructions as taught by Wang to the texture module of Rivard because these instructions are needed to render the graphics primitives to be displayed on the display device (col. 5 lines 43-45 and lines 63-67)." (Page 6, line 10 to Page 7, line 4)

> "However, the examiner interprets that Rivard teaches to send a request to DRAM via a texture module, and based on this request, DRAM sends back data and it's associated information to the texture module (see the arguments above for details). Nonetheless, Rivard does not explicitly state that the data and it's associated instructions are returned by DRAM, though the data has it's associated functions." (Page 8, lines 4-8)

First, appellant notes that the Examiner's arguments on Page 7, lines 5-21; and Page 8, lines 9-20 in the Office Action mailed 09/20/2007 are substantially similar to the arguments made by the

Examiner on Page 6, line 10 to Page 7, line 4 excerpted above. Further, on Page 7, line 21 the Examiner has additionally citied "*In re Lockhart*, 90 USPQ (CCPA 1951)." Second, appellant has clearly responded to such arguments above regarding how the excerpts repeatedly relied upon by the Examiner fail to meet or suggest appellant's claimed "receiving instructions from the video memory in response to the instruction request for storage in the cache in the graphics pipeline" (emphasis added), as claimed by appellant.

Further, appellant respectfully asserts that the excerpts from Rivard relied upon by the Examiner merely teach that "[m]emory request generator 1020 generates memory requests for all misses… and forwards the requests to DRAM 655 for information retrieval" such that "DRAM 655 returns the memory data on bus 660 to cache data store and memory data resolver 1030" (Col. 6, lines 50-55 – emphasis added). However, Rivard's disclosure that memory requests are forwarded to DRAM for information retrieval, where the DRAM returns the memory data, fails to support the Examiner's allegation that "Rivard further teaches receiving instructions from the video memory in response to the instruction request utilizing the texture module in the graphics pipeline" (emphasis added). Clearly, forwarding memory requests to DRAM for information retrieval, as in Rivard, simply fails to specifically teach "receiving instructions from the video memory in response to the instruction request for storage in the cache in the graphics pipeline" (emphasis added), as claimed by appellant.

In addition, appellant respectfully asserts that the excerpts from Rivard relied upon by the Examiner teach that "[b]ecause memory request generator 1020 is between cache tag blocks 1010, 1015 and cache data store 1030, generator 1020 can perform DRAM 655 memory requests before the address and instruction information reach cache data store and memory data resolver 1030" (Col. 6, lines 62-66 – emphasis added). Furthermore, Wang teaches "a 3D graphics subunit 109 for executing a series of display instructions found within a display list stored in computer memory" and that "[m]any of the polygon display instructions include texture data to be displayed within the polygon" (Col. 5, lines 40-48 – emphasis added). However, the mere disclosure of memory request generator 1020 performing DRAM memory requests before the instruction information reaches the cache data store, as in Rivard, in addition to the disclosure of executing display instructions that may include texture data, as in Wang, simply fails to suggest "receiving instructions from the video memory in response to the instruction request for storage

in the cache in the graphics pipeline" (emphasis added), as claimed by appellant. Clearly, performing memory requests before instruction information reaches the cache data store, as in Rivard, in addition to the disclosure of executing display instructions that may include texture data, as in Wang, simply fails to even suggest an "instruction request," much less "receiving instructions from the video memory in response to the instruction request for storage in the cache in the graphics pipeline" (emphasis added), as claimed by appellant.

Further, appellant respectfully asserts that Rivard's disclosure of a "graphics accelerator system 635 [that] includes pipeline latency elements 1025 to coordinate arrival of the memory data and of the associated instructions at cache data store and memory data resolver 1030" (Col. 7, lines 4-7) clearly *teaches away* from the Examiner's allegation that it would be obvious to "have the memory return instructions as taught by Wang to the texture module of Rivard because these instructions are needed to render the graphics primitives to be displayed on the display device," as noted by the Examiner, since the "pipeline latency elements... coordinate arrival of the memory data and... the associated instructions" (emphasis added). Clearly, as argued above, it would not be obvious from the teachings of Rivard and Wang to "receiv[e] instructions from the video memory in response to the instruction request for storage in the cache in the graphics pipeline" (emphasis added), as claimed by appellant.

Still yet, appellant agrees with the Examiner's statement that "Rivard does not explicitly state that the data and it's associated instructions are returned by DRAM." For example, appellant asserts that the figures and excerpts from Rivard relied upon by the Examiner teach that texel cache system 650 "includes cache tags 1010, cache tags 1015, a memory request generator 1020, pipeline latency elements 1025 and cache data store and memory data resolver 1030" (see Figure 10; Col. 6, lines 22-26 – emphasis added) where the "[m]emory request generator 1020 generates memory requests for all misses... and forwards the requests to DRAM 655 for information retrieval" (Col. 6, lines 50-53 – emphasis added). Clearly, a texel cache system 650 including a memory request generator 1020 that generates memory requests that are forwarded to DRAM 655 for information retrieval, as in Rivard, simply fails to suggest "receiving instructions from the video memory in response to the instruction request for storage in the cache in the graphics pipeline" (emphasis added), as claimed by appellant.

Again, appellant respectfully asserts that at least the first and third elements of the *prima facie* case of obviousness have not been met, since it would be *unobvious* to combine the references, as noted above, and the prior art excerpts, as relied upon by the Examiner, fail to teach or suggest all of the claim limitations, as noted above.

Group #5: Claim 18

With respect to Claim 18, the Examiner has relied on Col. 5, lines 43-67; and Col. 6, lines 1-47 from Wang to make a prior art showing of appellant's claimed technique "wherein a complete instruction set is received in response to the instruction request." Specifically, the Examiner has argued that "Wang… teaches a graphics subunit (texture module) of a graphics hardware system executing a series of display instructions (set of instructions) stored in computer memory" and that "[t]he graphics subunit receives display instructions including texture data based on the supplied data and control signals."

Appellant respectfully disagrees and asserts that Wang merely discloses that "[t]he graphics hardware system 108 contains a 3D graphics subunit 109 for executing a series of display instructions found within a display list stored in computer memory." where "[t]he display list generally contains instructions regarding the rendering of several graphic primitives, e.g., individual points, lines, polygons, fills, BIT BLTs (bit block transfers), textures, etc." (Col. 5, lines 40-46 – emphasis added). However, only generally disclosing the graphics subunit executing a series of display instructions found within a display list stored in computer memory, as in Wang, does not specifically teach that "a complete instruction set is received in response to the instruction request," where the "instructions [are received] from the video memory" (emphasis added), in the context as claimed by appellant (see Claim 1 for context).

In fact, appellant notes that the Examiner has relied on the same disclosure in Wang of a "series of display instructions" to meet both appellant's claimed "complete instruction set [that] is received in response to the instruction request" (see Claim 18) and "partial instruction set [that] is received in response to the instruction request" (see Claim 19), as claimed. Clearly, a generally disclosure of a series of instructions, as in Wang, simply cannot meet appellant's

claimed "complete instruction set" (Claim 18) and "partial instruction set" (Claim 19), as claimed.

Furthermore, in the Office Action mailed 09/20/2007, on Page 11, line 10 to Page 12, line 8, the Examiner has argued the following:

> "Regarding claim 18, Rivard does not explicitly teach a complete instruction set is received in response to the instruction request. However, Wang (col. 5 lines 43-67, col. 6 lines 1-47) teaches a graphics subunit (texture module) of a graphics hardware system executing a series of display instructions (set of instructions) stored in computer memory (Fig. 1, col. 5 lines 38-67; the computer memory includes the local frame buffer memory, which corresponds to the DRAM; based on the control signals send by the graphics hardware system, the display list stored in the frame buffer returns the polygon display instructions that include the texture data, so that the graphics subunit executes the display instructions using this data). The graphics subunit receives display instructions including texture data based on the supplied data and control signals (the examiner interprets that the claims do not specify how many instructions are needed to make a set complete; the examiner interprets that if the required operation can be performed from just the one received instruction, then it is considered as a complete set of instruction; the single display instruction received by the graphics from the sub-routine process in response to the control signals correspond to a complete set of instruction; the instruction in the sub-routine is executed by the graphics subunit for rendering the concerned graphics primitive, and therefore it is considered as a complete set of instruction). Therefore, it would have been obvious to one of ordinary skill in art at the time of present invention to have the memory return a complete set of instructions as taught by Wang to the texture module of Rivard because this instruction set is needed to render the concerned graphics primitive (col. 5 lines 43-45 and lines 63-67)." (Page 11, line 10 to Page 12, line 8)

Appellant agrees with the Examiner's argument that "Rivard does not explicitly teach a complete instruction set is received in response to the instruction request." Further, appellant respectfully disagrees with the Examiner's arguments with respect to Wang and asserts that the excerpts from Wang relied upon by the Examiner merely disclose that "[t]he graphics hardware system 108 contains a 3D graphics subunit 109 for executing a series of display instructions found within a display list stored in computer memory" where "[t]he display list generally contains instructions regarding the rendering of several graphic primitives, e.g., individual points, lines, polygons, fills, BIT BLTs (bit block transfers), textures, etc." (Col. 5, lines 40-46 – emphasis added). Further, the excerpts disclose that "[m]any of the polygon display instructions include texture data to be displayed within the polygon" (Col. 5, lines 46-48).

However, appellant asserts that Wang's disclosure of "a 3D graphics subunit 109 for executing a series of display instructions" (emphasis added), clearly fails to support the Examiner's allegation that "if the required operation can be performed from just the one received instruction, then it is considered as a complete set of instruction" (emphasis added). Clearly, a series of display instructions fails to support "just the one received instruction" (emphasis added), as alleged by the Examiner. Furthermore, Wang's disclosure of a series of display instructions simply fails to meet appellant's claimed "complete instruction set," much less disclose that "a complete instruction set is received in response to the instruction request," where the "instructions [are received] from the video memory" (emphasis added), in the context as claimed by appellant (see Claim 1 for context). Therefore, appellant asserts that it would not have been obvious to "have the memory return a complete set of instructions as taught by Wang to the texture module of Rivard because this instruction set is needed to render the concerned graphics primitive," as alleged by the Examiner.

Again, appellant respectfully asserts that at least the first and third elements of the *prima facie* case of obviousness have not been met, since it would be *unobvious* to combine the references, as noted above, and the prior art excerpts, as relied upon by the Examiner, fail to teach or suggest all of the claim limitations, as noted above.

Group #6: Claims 19 and 20

In addition, with respect to Claim 19, the Examiner has relied on Col. 5, lines 43-67; and Col. 6, lines 1-47 from Wang to make a prior art showing of appellant's claimed technique "wherein a partial instruction set is received in response to the instruction request." Specifically, the Examiner has argued that "Wang... teaches a graphics subunit (texture module) of a graphics hardware system executing a series of display instructions (set of instructions) stored in computer memory" and that "[t]he graphics subunit receives display instructions including texture data based on the supplied data and control signals."

Appellant respectfully disagrees and asserts that Wang merely discloses that "[t]he graphics hardware system 108 contains a 3D graphics subunit 109 for executing a series of display instructions found within a display list stored in computer memory" where "[t]he display list

generally contains instructions regarding the rendering of several graphic primitives, e.g., individual points, lines, polygons, fills, BIT BLTs (bit block transfers), textures, etc." (Col. 5, lines 40-46 – emphasis added). However, only generally disclosing the graphics subunit executing a series of display instructions found within a display list stored in computer memory, as in Wang, does not specifically teach that "a partial instruction set is received in response to the instruction request," where the "instructions [are received] from the video memory" (emphasis added), in the context as claimed by appellant (see Claim 1 for context).

In fact, appellant notes that the Examiner has relied on the same disclosure in Wang of a "series of display instructions" to meet both appellant's claimed "complete instruction set [that] is received in response to the instruction request" (see Claim 18) and "partial instruction set [that] is received in response to the instruction request" (see Claim 19), as claimed. Clearly, a generally disclosure of a series of instructions, as in Wang, simply cannot meet appellant's claimed "complete instruction set" (Claim 18) and "partial instruction set" (Claim 19), as claimed.

Furthermore, in the Office Action mailed 09/20/2007, Page 12, line 12 to Page 13, line 9, the Examiner has argued the following:

> "Regarding claims 19, Rivard does not explicitly teach a partial instruction set is received in response to the instruction request. However, Wang (col. 5 lines 43-67, col. 6 lines 1-47) teaches a graphics subunit (texture module) of a graphics hardware system executing a series of display instructions (set of instructions) stored in computer memory. The graphics subunit receives display instructions including texture data based on the supplied data and control signals (the examiner interprets that the claims do not specify how many instructions are needed to make a set complete or partial; the examiner interprets that if the required operation can be performed from just the one received instruction, then it is considered as a complete set of instruction; however, the rendering process might need other instructions for rendering other primitives, and in this case, the single received instruction is considered as a partial set of instruction; the single display instruction received by the graphics from the sub-routine process in response to the control signals correspond to a partial set of instruction; this instruction in the sub-routine is executed by the graphics subunit for rendering other primitives, and therefore overall this single instruction is considered as a partial set of instruction). Therefore, it would have been obvious to one of ordinary skill in art at the time of present invention to have the memory return instructions as taught by Wang to the texture module of Rivard because these instructions are needed to render the graphics primitives to be displayed on the display device (col. 5 lines 43-45 and lines 63-67)." (Page 12, line 12 to Page 13, line 9)

Appellant agrees with the Examiner's argument that "Rivard does not explicitly teach a partial instruction set is received in response to the instruction request." Further, appellant respectfully disagrees with the Examiner's arguments with respect to the Wang reference and asserts that the excerpts from Wang relied upon by the Examiner disclose that "[t]he graphics hardware system 108 contains a 3D graphics subunit 109 for executing a series of display instructions found within a display list stored in computer memory" where "[t]he display list generally contains instructions regarding the rendering of several graphic primitives, e.g., individual points, lines, polygons, fills, BIT BLTs (bit block transfers), textures, etc." (Col. 5, lines 40-46 – emphasis added). Further, the excerpts disclose that "[m]any of the polygon display instructions include texture data to be displayed within the polygon" (Col. 5, lines 46-48).

However, appellant asserts that Wang's disclosure of "a 3D graphics subunit 109 for executing a series of display instructions" (emphasis added), clearly fails to support the Examiner's allegation that "if the required operation can be performed from just the one received instruction, then it is considered as... a partial set of instruction" (emphasis added). Clearly, a series of display instructions fails to support "just the one received instruction," as alleged by the Examiner. Furthermore, Wang's disclosure of a series of display instructions simply fails to meet appellant's claimed "partial instruction set," much less disclose that "a partial instruction set is received in response to the instruction request," where the "instructions [are received] from the video memory" (emphasis added), in the context as claimed by appellant (see Claim 1 for context). Therefore, appellant asserts that it would not have been obvious to "have the memory return instructions as taught by Wang to the texture module of Rivard because these instructions are needed to render the graphics primitives to be displayed on the display device," as alleged by the Examiner.

Again, appellant respectfully asserts that at least the first and third elements of the *prima facie* case of obviousness have not been met, since it would be *unobvious* to combine the references, as noted above, and the prior art excerpts, as relied upon by the Examiner, fail to teach or suggest all of the claim limitations, as noted above.

Issue #2:

The Examiner has rejected Claims 13-17, 22, 23, and 29 under 35 U.S.C. 103(a) as being unpatentable over Rivard (U.S. Patent No. 5,987,567), in view of Wang (U.S. Patent No. 5,831,640), and further in view of Applicant Admitted Prior Art (AAPA).

Group #1: Claims 13-17, 22 and 23

Appellant respectfully asserts that such claims are not met by the prior art for the reasons argued with respect to Issue #1, Group #1.

Group #2: Claim 29

Furthermore, with respect to independent Claim 29, the Examiner has relied on Figures 6 and 10; Col. 4, lines 46-57; Col. 6, lines 45-67; and Col. 7, lines 1-10 from Rivard, along with Fig. 3; and Page 5, lines 24-31 from AAPA to make a prior art showing of appellant's claimed "sending an instruction request to video memory, where a texture module coupled to the shader module sends the instruction request to the video memory."

Appellant respectfully asserts that the figures relied on by the Examiner only generally illustrate a computer system and a block diagram detailing graphics accelerator system showing the pipeline latency elements 1025. In addition, in Col. 7, lines 4-7, Rivard teaches that "graphics accelerator system 635 includes pipeline latency elements 1025 to coordinate arrival of the memory data and of the associated instructions at cache data store and memory data resolver 1030" (emphasis added). Furthermore, Rivard teaches that "[m]emory request generator 1020 generates memory requests for all misses (up to four for bi-linear sampling and up to eight for tri-linear sampling), and forwards the requests..." (Col. 6, lines 50 – 52 emphasis added). However, disclosing that memory requests are generated for all misses, as in Rivard, fails to suggest "sending an instruction request to video memory, where a texture module coupled to the shader module sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

In addition, appellant respectfully asserts that Fig. 3 from AAPA as relied upon by the Examiner merely discloses that "the texels resulting from one texture lookup can influence the location of the texels in a subsequent texture lookup" (Page 4, lines 30-31 – emphasis added). However, the mere disclosure of texels resulting from a texture lookup fail to even suggest "sending an instruction request to video memory, where a texture module coupled to the shader module sends the instruction request to the video memory" (emphasis added), as claimed by appellant. Clearly, retrieving texels from a texture lookup fails to meet "an instruction request," in the manner as claimed by appellant.

In the Office Action mailed 03/20/2007, the Examiner has argued that "Rivard teaches sending an instruction request to video memory, where the texture module sends the instruction to the video memory." Appellant disagrees with the Examiner's arguments and asserts that, for substantially the same reasons as those argued above with respect to at least some of the independent claims, Rivard fails to even suggest "sending an instruction request," as appellant claims. For example, appellant respectfully points out that Rivard merely teaches that "the memory request generator is coupled between the cache tag block and the cache data store for performing a memory request before an address and instruction information associated with the needed texels reach the cache data store" (Col. 10, lines 48-52 - emphasis added). Clearly, performing a memory request before the address and instruction information reach the cache data store, as in Rivard (see excerpts above), fails to teach "sending an instruction request" (emphasis added), as claimed by appellant.

Furthermore, with respect to Claim 29 and the Examiner's reliance upon the rejection of Claim 1, the Examiner has argued that "the definition of instruction provided by dictionary.com, which defines instructions as a command given to a computer to carry out a particular operation and can contain data to be used in the operation; here DRAM sends memory data based on the memory requests/read requests from the texture module." Further, the Examiner has argued that "[m]emory requests/read requests act as an instruction to DRAM, which performs a particular function based on the request."

Appellant respectfully disagrees with the Examiner's arguments and asserts that the dictionary.com reference provided by the Examiner merely defines an instruction as "a command

given to a computer to carry out a particular operation." Appellant respectfully asserts that Rivard merely discloses that the "cache tag blocks 1010 and 1015 forward a cache write address to memory request generator 1020" which "generates memory requests for all misses .. and forwards the requests to DRAM 655 for information retrieval" (Col. 6, lines 33-36, and lines 48-53 – emphasis added). However, the disclosure of the memory request generator generating a memory request and forwarding the request to DRAM for information retrieval, as in Rivard, simply fails to specifically suggest "sending an instruction request to video memory, where a texture module coupled to the shader module sends the instruction request to the video memory" (emphasis added), as claimed by appellant. Clearly, generating and forwarding a memory request to DRAM, as in Rivard, fails to suggest "sending an instruction request" (emphasis added), in the manner as claimed by appellant.

In the Office Acton mailed 09/20/2007, on Page 2, line 19 to Page 3, line 19; and Page 4, lines 3-7, the Examiner has argued the following:

> "However, the examiner interprets that Rivard (Fig. 6, Fig. 10, col. 4 lines 46-57, col. 6 lines 45-67, col. 7 lines 1-10) teaches sending an instruction request to video memory, where a texture module in a graphics pipeline sends the instruction request to the video memory (a texture mapping stage sends information to texel cache system; the texture mapping stage and the texel cache system together are considered as texture module, so the information is passed between the components of the texture module; texel cache system comprising the memory request generator, pipeline latency elements, and cache data store and memory data resolver forwards the memory requests/read requests to DRAM for information retrieval. [T]he examiner interprets that Rivard discloses sending an instruction request to video memory by generating memory request for all misses, and forwards the requests to DRAM for information retrieval (Fig. 6, Fig. 10, col. 4 lines 46-57, col. 6 lines 45-67, col. 7 lines 1-10). It is interpreted that an instruction request could be considered as merely a request because it is basically a request that requires DRAM to perform necessary operation. The examiner also interprets that instructions stored in DRAM that are requested are basically stored as data. It is further interpreted that memory requests/read requests act as an instruction to DRAM based on which DRAM performs a particular function. Memory requests/read requests for all misses that are forwarded to DRAM are instruction requests based on which DRAM sends back information; the examiner relies on the definition of instruction provided by dictionary.com, which defines instructions as a command given to a computer to carry out a particular operation and can contain data to be used in the operation; here DRAM sends memory based on the memory requests/read requests from the texture module." (Page 2, line 19 to Page 3, line 19)

> "In response to [appellant's] arguments against the references individually, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986)." (Page 4, lines 3-7)

First, appellant respectfully asserts that the Examiner's arguments on Page 4, line 8 to Page 5, line 8 in the Office Action mailed 09/20/2007 are substantially similar to the arguments made by the Examiner on Page 2, line 19 to Page 3, line 19 excerpted above. Second, appellant has clearly responded to such arguments above regarding how the excerpts repeatedly relied upon by the Examiner fail to meet or suggest appellant's claimed "sending an instruction request to video memory, where a texture module coupled to the shader module sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

Further, appellant respectfully disagrees with the Examiner's arguments and asserts that the figures and excerpts from Rivard relied upon by the Examiner merely disclose that "the texture mode indicates a 16-bit per pixel texture lookup" in addition to "satisfy[ing] two lookup requests with a single read request" (Col. 6, lines 56-61 – emphasis added). Additionally, the excerpts from Rivard teach that "pipeline latency elements 1025… coordinate arrival of the memory data and of the associated instructions" (Col. 7, lines 3-7 – emphasis added) such that "the memory request generator… perform[s] a memory request before an address and instruction information associated with the needed texels reach the cache data store" (Col. 10, lines 48-52 – emphasis added). Clearly, the texture mode indicating a texture lookup, in addition to the memory request generator performing a memory request, as in Rivard, simply fails to support the Examiner's allegation that "a texture module… sends the instruction request to the video memory" (emphasis added), and especially does not teach "sending an instruction request to video memory, where a texture module coupled to the shader module sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

In addition, Rivard's disclosure that the "cache tag blocks 1010 and 1015 forward a cache write address to memory request generator 1020," which "generates memory requests for all misses… and forwards the requests to DRAM 655 for information retrieval" (Col. 6, lines 48-53 – emphasis added). However, the disclosure of the memory request generator generating a memory request and forwarding the request to DRAM for information retrieval, as in Rivard,

fails to support the Examiner's allegation "that Rivard discloses sending an instruction request to video memory by generating memory request for all misses, and forwards the requests to DRAM for information retrieval" (emphasis added). Clearly, generating a memory request and forwarding the request to DRAM for information retrieval, as in Rivard, simply fails to specifically teach "sending an instruction request to video memory, where a texture module coupled to the shader module sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

Still yet, appellant respectfully asserts that the excerpts from Rivard relied upon by the Examiner disclose that the "[g]raphics application program 670 transfers graphical information from data storage 625 into DRAM 655 for local storage" (Col. 4, lines 46-48 – emphasis added). Additionally, the excerpts disclose that "[i]f a hit occurs, then cache tag blocks 1010 and 1015 forward a cache read address through memory request generator 1020 and through pipeline latency elements 1025 to cache data store 1030" such that "[c]ache data store 1030 responsively outputs on lines 649 the texture values cached in the read location" (Col. 6, lines 42-47 – emphasis added). Clearly, Rivard is disclosing the transfer of graphical information into DRAM and outputting cached texture values, which simply fails to support the Examiner's allegation that "instructions stored in DRAM that are requested are basically stored as data" (emphasis added). Therefore, the disclosure of transferring graphical information into DRAM and outputting cached texture values, as in Rivard, simply fails to suggest "sending an instruction request to video memory, where a texture module coupled to the shader module sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

Furthermore, appellant again asserts that the dictionary.com reference provided by the Examiner merely defines an instruction as "a command given to a computer to carry out a particular operation." In addition, the excerpts from Rivard disclose that the "memory request generator 1020 generates memory requests for all misses... and forwards the requests to DRAM 655 for information retrieval" (Col. 6, lines 50-53 – emphasis added). However, generating and forwarding memory requests to DRAM for information retrieval, as in Rivard, fails to support the Examiner's allegation that "[m]emory requests/read requests for all misses that are forwarded to DRAM are instruction requests based on which DRAM sends back information" (emphasis added). Clearly, generating and forwarding memory requests to DRAM for information

retrieval, as in Rivard, simply fails to specifically teach "sending an instruction request to video memory, where a texture module coupled to the shader module sends the instruction request to the video memory" (emphasis added), as claimed by appellant. Appellant emphasizes that the memory request of Rivard fails to meet appellant's claimed "instruction request," as claimed by appellant.

Still yet, appellant respectfully disagrees with the Examiner's argument that "it would have been obvious to a person of ordinary skill in the art at the time of the invention to incorporate the shading module of AAPA into the texture module of Rivard and Wang because combination of shading module and texture module would enable a shading function of the graphics pipeline." Clearly, the excerpts from AAPA, Rivard, and Wang relied upon by the Examiner fail to suggest "sending an instruction request to video memory, where a texture module coupled to the shader module sends the instruction request to the video memory" (emphasis added), as claimed by appellant.

Furthermore, in the Office Action mailed 09/20/2007, in paragraph 15 on page 10, the Examiner has argued the following:

> "In response to [appellant's] arguments against references individually, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986).
>
> In this case, the examiner interprets that Rivard, in view of Wang teaches to send an instruction request to memory utilizing a texture module (see refer to the arguments above for details).
>
> Although the combination of Rivard and Wang teach a method and system for retrieving instructions from memory utilizing a texture module in a graphics pipeline, they do not explicitly teach controlling the texture module utilizing a shader module coupled thereto. However, AAPA teaches controlling the texture module utilizing a shader module coupled thereto (Fig. 3; shader module is also coupled to the rasterizer module). Therefore, it would have been obvious to a person of ordinary skill in the art at the time of the invention to incorporate the shading module of AAPA into the texture module of Rivard and Wang because combination of shading module and texture module would enable a shading function of the graphics pipeline."

Appellant respectfully disagrees.

In addition, with respect to Claim 29, the Examiner has relied on Figures 6 and 10; Col. 4, lines 46-57; Col. 6, lines 45-67; and Col. 7, lines 1-10 from Rivard, along with Fig. 3; and Page 5, lines 24-31 from AAPA to make a prior art showing of appellant's claimed "receiving additional instructions from the video memory in response to the instruction request utilizing the texture module."

Specifically, the Examiner has argued that "Rivard... teaches that DRAM returns memory data to cache data store and memory data resolver component of texel cache system, which further sends this information to the texture mapping stage." Further, the Examiner has argued that "the texture mapping stage and the texel cache system together are considered as texture module, so the information is passed between the components of the texture module." In addition, the Examiner has argued that "[t]he texture mapping stage as shown in Fig. 6 also receives data (instructions) from the rasterizer module of the pipeline, and thus the instructions received from the DRAM via the texel cache system are considered to be the additional instructions."

Appellant respectfully disagrees with the Examiner's arguments and asserts that Rivard merely discloses that "DRAM 655 returns the memory data on bus 660 to cache data store and memory data resolver 1030, which stores the memory data at the write address" (Col. 6, lines 53-56 – emphasis added). However, merely storing memory data at the write address, as in Rivard, simply fails to suggest "receiving additional instructions from the video memory in response to the instruction request utilizing the texture module" (emphasis added), as claimed by appellant. Clearly, storing memory data, as in Rivard, fails to suggest "receiving additional instructions," in the manner as claimed by appellant.

Second, in response to the Examiner's argument that "the texture mapping stage and the texel cache system together are considered as texture module," appellant respectfully disagrees and asserts that Rivard merely teaches that "[g]raphics accelerator system 635 includes graphics pipeline stages 640 including a texture mapping stage 645 for mapping texture information to the graphics information received from graphics application program 670 and for maintaining the texel information in a texel cache system 650" where the "[t]exture mapping block 645 sends

information via bus 647 to texel cache system 650, and texel cache system 650 sends information via bus 649 back to texture mapping block 645" (Col. 4, lines 49-57 – emphasis added). Clearly, the texture mapping stage and separate texture cache system, as in Rivard, fails to suggest "receiving additional instructions from the video memory in response to the instruction request utilizing the texture module" (emphasis added), as claimed by appellant.

Third, in response to the Examiner's argument that "Fig. 6 also receives data (instructions) from the rasterizer module of the pipeline, and thus the instructions received from the DRAM via the texel cache system are considered to be the additional instructions," appellant respectfully disagrees and asserts that Rivard merely teaches that "the memory request generator is coupled between the cache tag block and the cache data store for performing a memory request before an address and instruction information associated with the needed texels reach the cache data store" (emphasis added). Clearly, coordinating the arrival of "memory data", where "the data is conveniently aligned in the texture map" (emphasis added), as in Rivard (see excerpts above), fails to teach "receiving additional instructions" (emphasis added), as claimed by appellant. In addition, appellant respectfully asserts that simply nowhere in the description of Figure 6 is there any disclosure that "Fig. 6 also receives data (instructions) from the rasterizer module of the pipeline," as noted by the Examiner. Thus, Rivard simply fails to meet appellant's claimed "receiving additional instructions... utilizing the texture module," as claimed.

Furthermore, in the Office Action mailed 09/20/2007, on Page 7, lines 3-7; Page 5, lines 9-17; Page 6, line 10 to Page 7, line 4; and Page 8, lines 4-8, the Examiner has argued the following:

> "In response to [appellant's] arguments against the references individually, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986)." (Page 7, lines 3-7)
>
> "Rivard further teaches receiving instructions from the video memory in response to the instruction request utilizing the texture module in the graphics pipeline (it is interpreted that instructions stored in DRAM that are requested are basically stored as data, and Rivard teaches to send back data in some form in response to the memory request generated as a result of a miss; DRAM returns memory data to cache data store and memory data resolver component of texel cache system, which further sends this information to the texture mapping stage; the texture mapping stage and the texel cache

system together are considered as texture module, so the information/memory data is passed between the components of the texture module)." (Page 5, lines 9-17)

"The examiner further interprets that Rivard also teaches the arrival of memory data and associated instructions at the cache data store and memory data resolver component of the text cache system is coordinated by pipeline latency elements (col. 7 lines 4-7; memory data has associated instructions are returned by DRAM, though the data has it's associated instructions). Therefore, the examiner brings in the Wang reference that suggests a graphics subunit (texture module) in a graphics hardware system that supplies data and control signals to local frame buffer memory for executing a series of display instructions (Fig. 1, col. 5 lines 38-67; the computer memory includes the local frame buffer memory, which corresponds to the DRAM; based on the control signals send by the graphics hardware system, the frame buffer returns the polygon display instructions that includes the texture data, so that the graphics subunit executes the display instructions using this data). Therefore, it would have been obvious to one of ordinary skill in art at the time of present invention to have the memory return instructions as taught by Wang to the texture module of Rivard because these instructions are needed to render the graphics primitives to be displayed on the display device (col. 5 lines 43-45 and lines 63-67)." (Page 6, line 10 to Page 7, line 4)

"However, the examiner interprets that Rivard teaches to send a request to DRAM via a texture module, and based on this request, DRAM sends back data and it's associated information to the texture module (see the arguments above for details). Nonetheless, Rivard does not explicitly state that the data and it's associated instructions are returned by DRAM, though the data has it's associated functions." (Page 8, lines 4-8)

First, appellant notes that the Examiner's arguments on Page 7, lines 5-21; and Page 8, lines 9-20 in the Office Action mailed 09/20/2007 are substantially similar to the arguments made by the Examiner on Page 6, line 10 to Page 7, line 4 excerpted above. Further, on Page 7, line 21 the Examiner has additionally citied "*In re Lockhart*, 90 USPQ (CCPA 1951)." Second, appellant has clearly responded to such arguments above regarding how the excerpts repeatedly relied upon by the Examiner fail to meet or suggest appellant's claimed "receiving additional instructions from the video memory in response to the instruction request utilizing the texture module" (emphasis added), as claimed by appellant.

Further, appellant respectfully asserts that the excerpts from Rivard relied upon by the Examiner merely teach that "[m]emory request generator 1020 generates memory requests for all misses… and forwards the requests to DRAM 655 for information retrieval" such that "DRAM 655 returns the memory data on bus 660 to cache data store and memory data resolver 1030" (Col. 6, lines 50-55 – emphasis added). However, Rivard's disclosure that memory requests are forwarded to DRAM for information retrieval, where the DRAM returns the memory data, fails to support the

Examiner's allegation that "Rivard further teaches receiving instructions from the video memory in response to the instruction request utilizing the texture module in the graphics pipeline" (emphasis added). Clearly, forwarding memory requests to DRAM for information retrieval, as in Rivard, simply fails to specifically teach "receiving additional instructions from the video memory in response to the instruction request utilizing the texture module" (emphasis added), as claimed by appellant.

In addition, appellant respectfully asserts that the excerpts from Rivard relied upon by the Examiner teach that "[b]ecause memory request generator 1020 is between cache tag blocks 1010, 1015 and cache data store 1030, generator 1020 can perform DRAM 655 memory requests before the address and instruction information reach cache data store and memory data resolver 1030" (Col. 6, lines 62-66 – emphasis added). Furthermore, Wang teaches "a 3D graphics subunit 109 for executing a series of display instructions found within a display list stored in computer memory" and that "[m]any of the polygon display instructions include texture data to be displayed within the polygon" (Col. 5, lines 40-48 – emphasis added). However, the mere disclosure of memory request generator 1020 performing DRAM memory requests before the instruction information reaches the cache data store, as in Rivard, in addition to the disclosure of executing display instructions that may include texture data, as in Wang, simply fails to suggest "receiving additional instructions from the video memory in response to the instruction request utilizing the texture module" (emphasis added), as claimed by appellant. Clearly, performing memory requests before instruction information reaches the cache data store, as in Rivard, in addition to the disclosure of executing display instructions that may include texture data, as in Wang, simply fails to even suggest an "instruction request," much less "receiving additional instructions from the video memory in response to the instruction request utilizing the texture module" (emphasis added), as claimed by appellant.

Further, appellant respectfully asserts that Rivard's disclosure of a "graphics accelerator system 635 [that] includes pipeline latency elements 1025 to coordinate arrival of the memory data and of the associated instructions at cache data store and memory data resolver 1030" (Col. 7, lines 4-7) clearly *teaches away* from the Examiner's allegation that it would be obvious to "have the memory return instructions as taught by Wang to the texture module of Rivard because these instructions are needed to render the graphics primitives to be displayed on the display device,"

as noted by the Examiner, since in Rivard the "pipeline latency elements... coordinate arrival of the memory data and... the associated instructions" (emphasis added). Clearly, as argued above, it would not be obvious from the teachings of Rivard and Wang to "receiv[e] additional instructions from the video memory in response to the instruction request utilizing the texture module" (emphasis added), as claimed by appellant.

Still yet, appellant agrees with the Examiner's statement that "Rivard does not explicitly state that the data and it's associated instructions are returned by DRAM." For example, appellant asserts that the figures and excerpts from Rivard relied upon by the Examiner teach that texel cache system 650 "includes cache tags 1010, cache tags 1015, a memory request generator 1020, pipeline latency elements 1025 and cache data store and memory data resolver 1030" (see Figure 10; Col. 6, lines 22-26 – emphasis added) where the "[m]emory request generator 1020 generates memory requests for all misses... and forwards the requests to DRAM 655 for information retrieval" (Col. 6, lines 50-53 – emphasis added). Clearly, a texel cache system 650 including a memory request generator 1020 that generates memory requests that are forwarded to DRAM 655 for information retrieval, as in Rivard, simply fails to suggest "receiving additional instructions from the video memory in response to the instruction request utilizing the texture module" (emphasis added), as claimed by appellant.

Furthermore, in the Office Action mailed 09/20/2007, on Page 5, line 18 to Page 6, line 9, the Examiner has argued the following:

> "Although Rivard discloses the limitations as stated above, Rivard does not explicitly teach to combine texture mapping stage and texel cache system to form a texture module. However, it would have been obvious to one of ordinary skill in art at the time of present invention to combine texture mapping stage and texel cache system of Rivard to work together as a texture module. The unity of diversity of parts would depend more upon the choice of the manufacturer, and the convenience and availability of the machines and tools necessary to construct the texture module, than on any inventive concept. One of ordinary skill in art, furthermore, would have expected [appellant's] invention to perform equally well with Rivard's reference that teaches to send and receive information/instruction from the texture mapping stage and texel cache system to the DRAM because using this components together will also result in sending and receiving instructions to and from DRAM. Therefore, it would have been obvious to one of ordinary skill in art at the time of present invention to modify Rivard to obtain the invention as specified in the claim." (Page 5, line 18 to Page 6, line 9)

First, appellant respectfully asserts that the Examiner's arguments on Page 7, lines 8-21 of the Office Action mailed 09/20/2007 are substantially similar to the arguments made by the Examiner on Page 5, line 18 to Page 6, line 9 excerpted above. Further, on Page 7, line 21 the Examiner has additionally citied "*In re Lockhart*, 90 USPQ (CCPA 1951)."

Further, appellant respectfully disagrees with the Examiner's arguments and asserts that it would not be obvious to "combine texture mapping stage and texel cache system of Rivard to work together as a texture module," as alleged by the Examiner. For example, Rivard teaches a "[g]raphics accelerator system 635 includes graphics pipeline stages 640 including a texture mapping stage 645 for mapping texture information to the graphics information received from graphics application program 670 and for maintaining the texel information in a texel cache system 650" such that "[t]exture mapping block 645 sends information via bus 647 to texel cache system 650, and texel cache system 650 sends information via bus 649 back to texture mapping block 645" (Col. 4, lines 49-57 – emphasis added). Clearly, Rivard teaches and suggests the use of graphic pipeline stages, such as a texture mapping block 645 and texel cache system 650, which simply fails to support the Examiner's allegation that it would be obvious to "combine texture mapping stage and texel cache system of Rivard to work together as a texture module." Therefore, Rivard's teachings of a separate texture mapping block 645 and texel cache system 650 simply fails to suggest "receiving additional instructions... in response to the instruction request utilizing the texture module" (emphasis added), as claimed.

Again, appellant respectfully asserts that at least the first and third elements of the *prima facie* case of obviousness have not been met, since it would be *unobvious* to combine the references, as noted above, and the prior art excerpts, as relied upon by the Examiner, fail to teach or suggest all of the claim limitations, as noted above.

In view of the remarks set forth hereinabove, all of the independent claims are deemed allowable, along with any claims depending therefrom.

VIII CLAIMS APPENDIX (37 C.F.R. § 41.37(c)(1)(viii))

The text of the claims involved in the appeal (along with associated status information) is set forth below:

1. (Previously Presented) A method for execution with a system including a tangible computer readable medium, the method for retrieving instructions from video memory utilizing a texture module in a graphics pipeline, comprising:
(a) sending an instruction request to video memory, where a texture module in a graphics pipeline sends the instruction request to the video memory; and
(b) receiving instructions from the video memory in response to the instruction request utilizing the texture module in the graphics pipeline.

2. (Previously Presented) The method as recited in claim 1, and further comprising sending a texture request to video memory utilizing the texture module in the graphics pipeline.

3. (Previously Presented) The method as recited in claim 2, and further comprising receiving texture information from the video memory in response to the texture request utilizing the texture module in the graphics pipeline.

4. (Previously Presented) The method as recited in claim 1, wherein the video memory includes a frame buffer.

5. (Previously Presented) The method as recited in claim 4, wherein the video memory includes direct random access memory (DRAM).

6. (Original) The method as recited in claim 3, wherein the instructions are adapted for controlling a texture environment module coupled to the texture module.

7. (Original) The method as recited in claim 6, wherein the instructions control the manner in which the texture environment module processes the texture information.

8. (Original) The method as recited in claim 1, and further comprising receiving initial instructions from a rasterizer module coupled to the texture module.

9. (Original) The method as recited in claim 8, wherein the initial instructions control at least the sending of the instruction request by the texture module.

10. (Original) The method as recited in claim 3, and further comprising temporarily storing the instructions and the texture information in cache.

11. (Original) The method as recited in claim 10, wherein the cache is resident on the texture module.

12. (Previously Presented) The method as recited in claim 3, wherein each piece of texture information and each of the instructions are of a similar size in the video memory.

13. (Original) The method as recited in claim 3, and further comprising controlling the texture module utilizing a shader module coupled thereto.

14. (Original) The method as recited in claim 13, wherein the shader module controls the sending of the instruction request and the texture request by the texture module.

15. (Original) The method as recited in claim 13, wherein the shader module processes a plurality of pixels with the texture information based on the instructions.

16. (Previously Presented) The method as recited in claim 15, wherein the shader module is capable of reusing the texture information in order to request further texture information from the video memory.

17. (Original) The method as recited in claim 15, and further comprising ceasing the processing upon the receipt of a terminate instruction.

18. (Original) The method as recited in claim 1, wherein a complete instruction set is received in response to the instruction request.

19. (Original) The method as recited in claim 1, wherein a partial instruction set is received in response to the instruction request.

20. (Original) The method as recited in claim 19, and further comprising repeating (a) – (b) in accordance with the instructions.

21. (Original) The method as recited in claim 1, wherein (a) – (b) are carried out in accordance with the instructions received in response to the instruction request.

22. (Original) The method as recited in claim 1, wherein the texture module is adapted for operating in a plurality of different modes.

23. (Original) The method as recited in claim 22, wherein the instructions are received in a predetermined one or more of the different modes.

24. (Previously Presented) A computer program product embodied on a tangible computer readable medium for retrieving instructions from video memory utilizing a texture module in a graphics pipeline, comprising:
(a) computer code for sending an instruction request to video memory, where a texture module in a graphics pipeline sends the instruction request to the video memory; and
(b) computer code for receiving instructions from the video memory in response to the instruction request utilizing the texture module in the graphics pipeline.

25. (Previously Presented) A system including a tangible computer readable medium, the system for retrieving instructions from video memory utilizing a texture module in a graphics pipeline, comprising:
(a) means for sending an instruction request to video memory, where a texture module in a graphics pipeline sends the instruction request to the video memory; and

(b) means for receiving instructions from the video memory in response to the instruction request.

26. (Previously Presented) A texture sub-system including a tangible computer readable medium, the texture sub-system for retrieving instructions from video memory and capable of carrying out a method, comprising:

(a) sending an instruction request to video memory, where a texture module sends the instruction request to the video memory; and

(b) receiving instructions from the video memory in response to the instruction request.

27. (Previously Presented) A data structure stored in a frame buffer of a graphics processor including a tangible computer readable medium, the data structure for allowing the retrieval of instructions, where a texture module coupled thereto sends the instruction request to video memory, the data structure comprising an instruction object stored in the frame buffer for being retrieved therefrom in response to the instruction request utilizing the texture module in the graphics processor.

28. (Previously Presented) A method for execution with a system including a tangible computer readable medium, the method for retrieving instructions from video memory, comprising:

(a) receiving a plurality of preliminary instructions from a rasterizer module utilizing a texture module coupled thereto;

(b) sending an instruction request to video memory, where the texture module sends the instruction request to the video memory;

(c) receiving additional instructions from the video memory in response to the instruction request utilizing the texture module;

(d) caching the additional instructions on the texture module;

(e) sending a texture request to video memory utilizing the texture module in accordance with the additional instructions;

(f) receiving texture information from the video memory in response to the texture request utilizing the texture module;

(g) caching the texture information on the texture module; and

(h) repeating (b) – (g) in accordance with the additional instructions.

29. (Previously Presented) A method for execution with a system including a tangible computer readable medium, the method for retrieving instructions from video memory, comprising:

(a) receiving a plurality of preliminary instructions from a rasterizer module utilizing a shader module coupled thereto;

(b) sending an instruction request to video memory, where a texture module coupled to the shader module sends the instruction request to the video memory;

(c) receiving additional instructions from the video memory in response to the instruction request utilizing the texture module;

(d) caching the additional instructions on the texture module;

(e) sending a texture request to video memory utilizing the texture module in accordance with the additional instructions;

(f) receiving texture information from the video memory in response to the texture request utilizing the texture module;

(g) caching the texture information on the texture module;

(h) processing a plurality of pixels with the texture information utilizing the shader module in accordance with the additional instructions;

(i) repeating (b) – (h) in accordance with the additional instructions; and

(j) outputting the processed pixels upon receipt of additional instructions that include a terminate instruction.

30. (Previously Presented) A method for execution with a system including a tangible computer readable medium, the method for retrieving instructions from video memory utilizing a cache in a graphics pipeline, comprising:

sending an instruction request to video memory in a graphics pipeline, where a cache in the graphics pipeline sends the instruction request to the video memory; and

receiving instructions from the video memory in response to the instruction request for storage in the cache in the graphics pipeline.

IX EVIDENCE APPENDIX (37 C.F.R. § 41.37(c)(1)(ix))

There is no such evidence.

X RELATED PROCEEDING APPENDIX (37 C.F.R. § 41.37(c)(1)(x))

Since no decision(s) has been rendered in such proceeding(s), no material is included in this Related Proceedings Appendix.

In the event a telephone conversation would expedite the prosecution of this application, the Examiner may reach the undersigned at (408) 971-2573. For payment of any additional fees due in connection with the filing of this paper, the Commissioner is authorized to charge such fees to Deposit Account No. 50-1351 (Order No. NVIDP064).

Respectfully submitted,

By: /KEVINZILKA/ Date: March 19, 2008
Kevin J. Zilka
Reg. No. 41,429

Zilka-Kotab, P.C.
P.O. Box 721120
San Jose, California 95172-1120
Telephone: (408) 971-2573
Facsimile: (408) 971-4660